UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Gregor Sušnik

Prototip nadzora in evidence dostopa s pomočjo pametnega telefona

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: viš. pred. dr. Igor Rožanc

Ljubljana, 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Sodoben nadzor dostopa v prostore zahteva elektronski sistem, ki pogosto temelji na specifičnih tehnologijah, ki zahtevajo uporabo posebnih kartic, kar ni poceni. Te rešitve je smiselno nadgraditi z evidentiranjem časa prisotnosti, s čemer se uporabnost še poveča.

V diplomski nalogi predlagajte elektronski sistem za nadzor in evidentiranje delovnega časa z uporabo NFC tehnologije, ki omogoča v ta namen uporabiti pametne telefone. Idejo sistematično razvijte v prototipno rešitev, ki obsega namizno aplikacijo za nastavitve in evidenco časa prisotnosti ter mobilno aplikacijo, s katero omogočate prepoznavanje identitete na NFC terminalih.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Gregor Sušnik, vpisna številka 63120102, avtor zaključnega dela z naslovom:

Prototip nadzora in evidence dostopa s pomočjo pametnega telefona

IZJAVLJAM

1. da sem pisno zaključno delo študija izdelal samostojno pod mentorstvom viš. pred. dr. Igorja Rožanca,
2. da je tiskana oblika pisnega zaključnega dela študija istovetna elektronski obliki pisnega zaključnega dela študija,
3. soglašam, da se elektronska oblika pisnega zaključnega dela študija uporabi za preverjanje podobnosti vsebine z drugimi deli s programsko opremo za preverjanje podobnosti vsebine, ki je povezana s študijskim informacijskim sistemom članice;,
4. da na UL neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve avtorskega dela v elektronski obliki, pravico reproduciranja ter pravico dajanja pisnega zaključnega dela študija na voljo javnosti na svetovnem spletu preko Repozitorija UL;
5. dovoljujem objavo svojih osebnih podatkov, ki so navedeni v pisnem zaključnem delu študija in tej izjavi, skupaj z objavo pisnega zaključnega dela študija.

V Ljubljani, dne 26. februarja 2016 Podpis študenta/-ke:

Zahvaljujem se mentorju viš. pred. dr. Igorju Rožancu za pomoč in nasvete pri diplomskem delu. Posebna zahvala gre tudi Evi Pišljar Suhadolc. Zahvaljujem pa se tudi prijateljem. Hvala pa tudi družinskim članom, ki so mi tekom študija stali ob strani in mi bili v podporo.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
AID	Application Identification	Identifikacijska številka aplikacije
API	Application Programming Interface	Vmesnik za namensko programiranje
APDU	Application Protocol Data Unit	Aplikacijski protokol podatkovne enote
CLS	Class	Razred
CPU	Central Processing Unit	Centralno procesna enota
ECMA	European Computer Manufacturers Association	Zasebna neprofitna standardizacijska organizacija za informacijske in komunikacijske sisteme
HCE	Host-based Card Emulation	Posnemanje pametne kartice na mobilnem telefonu
INS	Instruction	Ukaz
ISO/IEC	International Organization for Standardization / International Electrotechnical Commission	Mednarodna organizacija za standardizacijo / Mednarodna komisija za elektrotehniko
JDBC	Java Database Connectivity	Gonilnik za povezljivost z bazami podatkov Java
LAN	Local Area Network	Lokalno omrežje
NFC	Near Field Communication	Visokofrekvenčna komunikacijska tehnologija kratkega dosega
P2P	Peer-to-Peer	Infrastruktura omrežnega sistema
RFID	Radio-Frequency Identification	Radio frekvenčna identifikacija
WAN	Wide Area Network	Prostrano omrežje

Povzetek

Namen diplomskega dela je izdelati delujoč prototip aplikacije, katere glavna naloga je beleženje delovnega časa, vodenje evidence časa dela ter nadzor zaposlenih z uporabo kontrole dostopa, ki je izvedena s pomočjo pametnega telefona.

Kontrola dostopa je učinkovit sistem za izvajanje zagotavljanja varnosti fizičnih prostorov in digitalnih informacij pred nepooblaščenim dostopom. Kontrola dostopa se lahko nadgradi še v samodejno beleženje delovnega časa zaposlenih. V teoretičnem delu smo raziskali omenjena sistema in se na podlagi ugotovitev v naslednjem poglavju lotili idejne zasnove prototipa. Ideja je prerasla v konkretno analizo zahtev in izvedbo. Vsako zahtevo smo implementirali posamično. Z uporabo različnih programskih rešitev smo uspešno izdelali prototip sistema, ki s pomočjo pametnega telefona in namizne aplikacije ter NFC terminala beleži evidenco ur dela in omogoča kontrolo dostopa. Rezultat je delujoča aplikacija, ki je v zadnjem poglavju še preizkušena v produkcijskem okolju.

Ključne besede: registracija delovnega časa, kontrola dostopa, mobilna aplikacija, namizna aplikacija, pameten telefon

Abstract

The main aim of this thesis is to develop a functioning application prototype of a time-and-attendance system for automatic time and attendance registration, time and attendance management and for controlling employees' access, using smart phones.

The access control is an excellent solution for security management of physical rooms as well as digital information, with a purpose of preventing unauthorized access. Access control can be further upgraded into a time-and-attendance system. In the theoretical part of our thesis both systems are researched, resulting in a conceptual design of our prototype. A conceptual design advanced into requirements analysis. Every requirement was implemented separately. A prototype was created using different software solutions. Via mobile phone application, desktop application and NFC terminal, our prototype is able to run access control service and time attendance service. The result of this thesis is therefore a working prototype. In the last section, the prototype is tested and analysed in production environment.

Keywords: time attendance, access control, mobile application, desktop application, smart phone

Poglavje 1 Uvod

Kontrola dostopa in beleženje delovnega časa sta zelo pomembni aktivnosti vsakega podjetja. Prva se dotika najpomembnejše stvari, varnosti in varovanja prostorov pred nepooblaščenimi dostopi, druga pa močno olajša in na dolgi rok prihrani ogromne količine časa podjetju, s tem pa tudi denar. Sistemi, ki omogočajo omenjeni aktivnosti že obstajajo in temeljijo pretežno na RFID tehnologiji [6]. V diplomski nalogi želimo izdelati sistem, ki bo RFID tehnologijo zamenjal z NFC tehnologijo [6].

Z uporabo odprtokodnih programskih rešitev smo izdelali aplikacijo, ki implementira omenjeni aktivnosti. Aplikacija se sestoji iz namizne glavne aplikacije in mobilne aplikacije. Obe smo izdelali v programskem jeziku Java in temeljita na uporabi NFC tehnologije. Za komunikacijo med napravama smo uporabili NFC bralnik ACS122U [23]. Pri tem bi lahko uporabil tudi drug NFC bralnik, paziti smo morali le, da bralnik razume APDU [15] ukaze.

Glavna naloga bralnika je prenašanje sporočil med aplikacijama. Glavna aplikacija za pridobivanje in shranjevanje podatkov uporablja tudi podatkovno zbirko. To smo implementirali s pomočjo programskega paketa WAMP [14] in njegovega modula Phpmyadmin [25].

V diplomskem delu sta predstavljena sistema kontrole dostopa in evidence delovnega časa. Delo se nadaljuje z opisom uporabljenih metod, programov in knjižnic. Sledi razprava o idejni zasnovi, kjer je opisan nastanek ideje in razvoj prototipa. Sledi podrobnejši opis in razlage od začetne faze programiranja aplikacije do končnega - delujočega prototipa. Na koncu je opis predstavitve prototipa z vidika zaposlenega in administratorja. Ob zaključku je podan še lasten razmislek o opravljenem delu in možnosti nadaljnje nadgradnje in izboljšav prototipa.

Poglavje 2 Teoretična predstavitev sistemov

V teoretičnemu poglavju predstavimo kontrolo dostopa in evidenco delovnega časa. Sledijo ji opisi tehnologij in programskih rešitev, ki smo jih uporabili pri izdelavi prototipa.

2.1 Kontrola dostopa

Kontrola dostopa (angl. access control) [2] je danes prisotna skoraj v vsakem podjetju. Vsako podjetje strmi k varovanju pomembnih poslovnih prostorov in si želi imeti popoln nadzor in kontrolo, kdo vse ima dostop do pomembnih prostorov ali zaupnih informacij. Kontrola dostopa oz. modernejši izraz elektronska kontrola dostopa je eden izmed osnovnih sistemov za varovanje prostora pred nepooblaščenim vstopom in zagotavljanje računalniško podprtega varovanja določenega vstopnega območja v smislu pridobitve digitalne informacije. Z implementacijo in uvedbo celovitega sistema se močno zmanjša tveganje in izpostavljenost podjetja pred nepooblaščenim dostopom do informacij in ostalih pomembnih stvari. Glavna naloga sistema elektronske kontrole dostopa poskrbi, da imajo točno določene osebe v določenih časovnih intervalih, dostop do pomembnih informacij ali elektronsko varovanih prostorov. Celotni sistem sestavljata programska in strojna oprema.

2.1.1 Strojna oprema

Strojna oprema je namenjena bolj fizičnemu varovanju kot je recimo odobritev dostopa v nek prostor. Pri omejevanju dostopa [3] do pomembnih prostorov podjetja se pogosto uporablja električna ključavnica ali elektronski čitalec. Obe napravi se nahajata na sprednji strani vrat ter delujeta v navezi s programsko opremo sistema. Tako čitalec kot elektronska ključavnica za identifikacijo osebe uporabljata več vrst tehnologij. Starejša tehnologija (npr. uporaba klasičnih pametnih kartic) je v veliki meri še vedno aktualna. Zadnje čase je zelo izrazit trend nadomeščanja pametnih kartic z uporabo pametnih telefonov. Trend pa se premika tudi k implementaciji in razvoju naprednejših in varnejših tehnologij. Te temeljijo na uporabi biometričnih (angl. biometric) lastnosti osebe denimo prepoznavanje prstnega odtisa in podobno. Slednja predstavlja izredno varno kontrolo dostopa.

Oseba se mora za vstop v prostor identificirati. Pri uporabi klasične tehnologije pametnih kartic to stori tako, da približa pametno kartico ali kakšno drugo napravo za prepoznavanje identitete osebe čitalniku, ki je tipično nameščen poleg vrat. Pri uporabi naprednejših tehnologij se postavi pred senzor, da le-ta izvede postopek prepoznavanja biometričnih lastnosti.

2.1.1.1 Elementi strojne opreme

Slika 2.1 prikazuje tipične elemente kontrole dostopa, ki so:

- **Zapiralo vrat** se nahaja nad vrati in poskrbi, da se vrata ob odpiranju vrata zaprejo samodejno.
- **Magnetni trak** se nahaja na vrhu vrat in deluje v povezavi z električno ključavnico ali čitalcem. Sistemu izda opozorilo, če so bila vrata poskušana odpreti nasilno ali pa so bila vrata predolgo odprta.
- **Sirena** v primeru izdanega opozorila zvočno opozori na nepravilnosti.
- **Domofon** se nahaja na prednji strani vrat in omogoča komunikacijo med operaterjem in osebo. Pri tem operater ročno odpre vrata obiskovalcem s pomočjo stikala, ki se nahaja poleg domofona.
- **Video nadzorni sistem** poskrbi za še bolj učinkovito zagotavljanje varnosti, Zagotavlja, da se pri procesu identifikacije kontrole dostopa identificira pravi imetnik kartice ali druge naprave za preverjanje identifikacije osebe.

Slika 2.1: Prikaz elementov kontrole dostopa

2.1.2 Programska oprema

Povezovalni programski del sistema skrbi, da so vsi elementi pravilno logično sestavljeni v sistem. Signale naprav mora sprejeti, jih pravilno in hitro obdelati ter se na koncu ustrezno odzvati. Osebo, ki je zadolžena za administracijo sistema, mora obveščati o dogodkih, ki se dogajajo med izvajanjem elektronske kontrole dostopa. Prav tako mora programska rešitev skrbeti za hranjenje informacijskih podatkov celotnega sistema ter vseh podsistemih, ki so vključeni v podjetje kot celoto. Ti podatki morajo biti pooblaščenim osebam vedno na voljo. Podatki naj se hranijo v skladu z internim dogovorom podjetja. S tem mislimo na shranjevanje in dostop vseh podatkov zaposlenih, zaupne in tajne podatke sistema ali podjetja, podatke pravic in podobno. Pri dostopanju do informacij, ki so v digitalni obliki, mora programski del sistema zagotavljati programsko rešitev implementacije kontrole dostopa, ki na podlagi zahtevka osebe do zaupnih informacij odobri ali pa zavrne zahtevek. Tipičen je primer, ko zaposleni na svojem računalniku želi dostopati do zaupnih dokumentov, ki se v podjetju hranijo na posebnem strežniškem mestu. [4]

2.2 Registracija delovnega časa

Elektronski sistem za registracijo in vodenje evidenc delovnega časa (angl. time attendance) [1], [2], [5] je učinkovit in nepogrešljiv sistem za avtomatično spremljanje prihodov in odhodov zaposlenih. Sistem je precej učinkovitejši in hitrejši od zastarelega ročnega beleženja delovnega časa. Njegov namen je natančno beleženje prihodov in odhodov na delovno mesto, beleženje trajanja odsotnosti zaradi službenih ali privatnih zadev, beleženje trajanja malice, bolniške odsotnosti, koriščenja ur in dopusta. Sistem za vodenje evidence delovnega časa zaposlenih lahko deluje kot samostojna enota, ki poteka na ločenem sistemu podjetja, lahko pa je del sistema elektronske kontrole pristopa, pri čemer, se oseba evidentira enako kot pri kontroli pristopa z uporabo identifikacijskih sredstev kot so pametne kartice, pametni telefoni ali drugih metod identifikacije. Proces evidentiranja je potrebno podpreti tudi programsko.

Ker gre pogosto za specifične zahteve ne obstaja neka univerzalna rešitev, ki bi v popolnosti ugodila vsem podjetjem. Interni dogovori in kodeksi so različni od podjetja do podjetja, zato so specifične potrebe stvar medsebojnega sodelovanja med podjetjem in izvajalcem, kateri zagotovi implementacijo sistema po zahtevah podjetja. Na izbiro imamo ogromno podjetji, ki se ukvarjajo z implementacijo takšnih sistemov. Nekateri izmed teh podjetji so Špica [32], Četrta pot [33], Jantar [34] in še kopica ostalih. Vsem je skupno to, da ponujajo tako sistem za registracijo in evidentiranje delovnega časa kot tudi sistem za kontrolo pristopa. Podjetja nudijo splošno programsko opremo, za primere večjih podjetji pa ponujajo programske rešitve po meri.

Prihodnost kontrole dostopa [6], [8], [7] se kaže pri uporabi biometrije. Za svoje delovanje ne potrebuje kartic, ključev, pametnih telefonov ali drugih predmetov, s pomočjo katerih se izvede identifikacija. Biometrija ni prenosljiva.

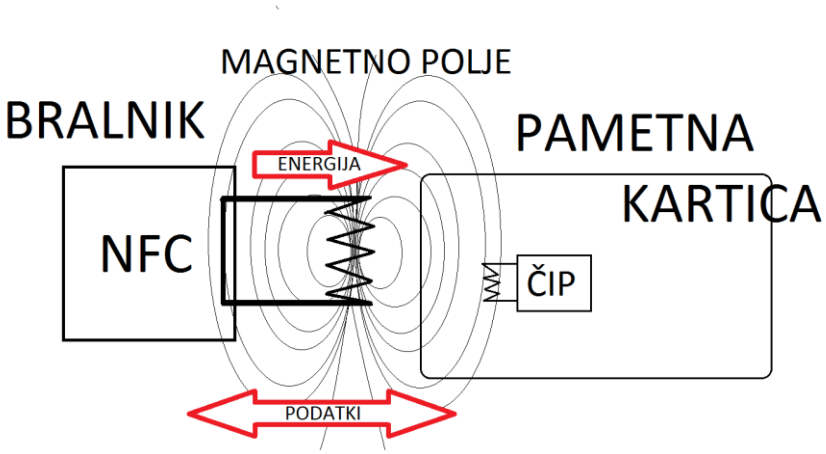
Danes je področje biometrije v polnem teku razvoja in integracije v naše življenje. Tako smo leta 2013 dobili pametni telefon na katerem je del področja biometrije, saj je moč prepoznati prstni odtis. S tem namenom lahko nadomestimo klasično uporabo gesel za odklepanje z biometrično metodo prepoznavanja prstnega odtisa lastnika telefona in odklenemo telefon ali pa izvedemo digitalni nakup. Področje biometrije za današnje čase omogoča identifikacijo na podlagi govora, obraza, oči, dlani, prstnih odtisov, lastnoročni podpis in podobno.

Biometrija sicer predstavlja najučinkovitejšo identifikacijo, vendar je sama implementacija tehnologije strojno zelo zahtevna in predvsem draga. Za doseganje enakih ciljev je ena od alternativ NFC tehnologija. Z njo praktično dobimo isto funkcionalnost kot pri ostalih možnih tehnologijah, s tem, da je NFC tehnologija cenejša in bistveno lažja za implementacijo.

2.3 NFC tehnologija

Visokofrekvenčna komunikacijska tehnologija na kratki razdalji [9], [10] (angl. Near Field Communication) ali krajše NFC tehnologija je naslednica radio frekvenčne identifikacije (angl. Radio Frequency Identification), krajše RFID. Lahko bi rekli, da gre za izpeljavo RFID tehnologije, pri čemer je oddaljenost med dvema NFC napravama manjša zaradi varnostnih razlogov. NFC deluje po principu radio-frekvenčnih signalov, ki jih prikazuje Slika 2.2.

Pametna kartica nima svojega lastnega napajanja ali vira energije, ima pa vgrajene tuljave. Ko je NFC bralnik pod napetostjo, s pomočjo svojih tuljav ustvari magnetno polje. Ko se pametna kartica približana bralniku in s tem magnetnemu polju, se s pomočjo tuljav v pametni kartici in magnetnega polja napaja – dobi energijo. Pri procesu nastane tudi nekaj sevanja, ki pa je zanemarljivo. Informacija je zapisana kot intenzivnost energije, ki se pretvarja iz ene v drugo napravo. Bralnik na podlagi intenzitete energije dekodira informacijo, ki jo pošlje pametna kartica nazaj.



Slika 2.2: Prikaz delovanja NFC tehnologije

Kot ISO/IEC standard je bil NFC odobren 8. decembra, 2003, kmalu zatem pa še kot ECMA standard. Pri uporabnosti NFC tehnologije je veliko pripomoglo neprofitno industrijsko združenje imenovano NFC Forum, ki so ga ustanovili Philips, Sony ter Nokia. Združenje se zavzema za razširjenost tehnologije, čim bolj preprosto uporabo in obravnava varnostne vidike tehnologije. Združenje je postavilo jasne standarde, ki morajo biti izpolnjeni pri načrtovanju različnih vrst naprav, ki želijo implementirati NFC. S tem je zagotovljeno, da lahko neodvisne naprave med seboj komunicirajo.

Kmalu za tem so leta 2006 definirali in razvili tako imenovane NFC značke. Slika 2.3 prikazuje uradne verzije logotipov. Gre za izredno majhno stvar, ki vsebuje informacije, ki jih je moč prebrati ob stiku z uporabo pametnega mobilnega telefona, ki ima podporo NFC tehnologije, ali katerimi drugimi napravami, ki podpirajo NFC. Tipično je značka izdelana za branje informacij, obstajajo pa tudi značke, na katere lahko zapišemo informacijo ali pa jo spremenimo.

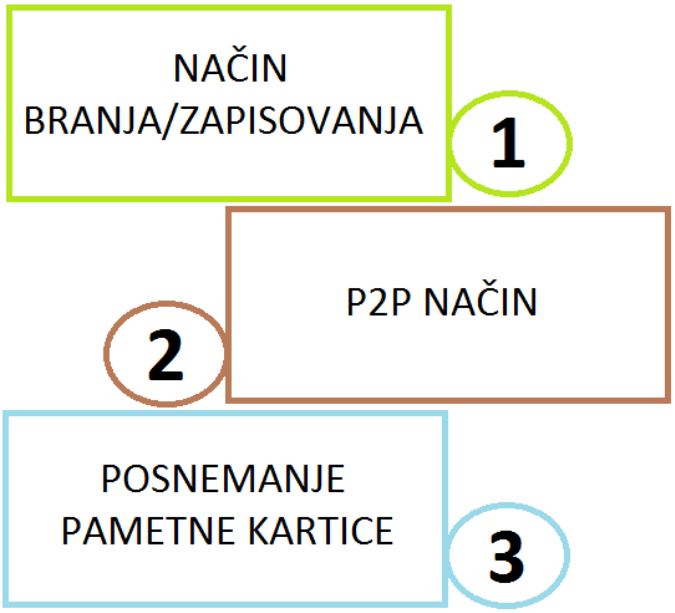
Slika 2.3: Uradne verzije logotipa NFC tehnologije.

Danes se NFC uporablja predvsem v mobilni industriji. Podjetja kot so Google, Apple in podobno. vse bolj strmijo k implementaciji NFC tehnologije za svoje storitve, denimo Google

Wallet [26] ter Apple Pay [27]. Najbolj razširjena uporaba je pri sistemih za beleženje delovnega časa zaposlenih, omogočanje dostopa zaposlenim v podjetju, uporablja se tudi v namene javnih prevozov. Vse več podjetji, predvsem banke, ki strmijo k brezkontaktnemu bančništvu na mobilnih napravah, pa strmi k razvoju in implementacij NFC tehnologije na mobilnih napravah, s katerim bi nadomestili klasični sistem brezkontaktnih kartic.

V primerjavi s tehnologijo Bluetooth [28], katere namen je prenos datotek brez povezljivosti fizičnega kabla med bližnjima napravama, je NFC izredno hiter pri sami vzpostavitvi komunikacije med dvema napravama, saj ni potrebe po nastavljanju komunikacije med eno in drugo napravo. Vse skupaj se zgodi takoj, ko sta napravi približani ena drugi.

Tehnologija Wi-Fi [29] je zelo podobna NFC tehnologiji, vendar je bila sprva zasnovana za povezovanje naprav znotraj LAN omrežja, danes pa tudi WAN omrežja. Prav tako Wi-Fi odpravlja potrebo po fizični povezljivosti naprav v omrežju in ima domet do 100 metrov.



Slika 2.4: Ponazoritev možnih načinov uporabe NFC tehnologije

Tehnologija, ki deluje v ozadju NFC, omogoča, da se naprava obnaša kot

- pasivna naprava ali
- aktivna naprava

V primeru pasivnega načina naprava služi le kot značka in hrani neko informacijo (primer številka 1 na sliki Slika 2.4). Pametni telefoni z ustrezno aplikacijo ali druge naprave lahko znački preberejo informacijo ali pa jo vanjo zapišejo.

V aktivnem načinu naprava deluje kot oponašalec pametne kartice ali kot komunikator še z eno NFC napravo, kot ponazarjata primera številka 2 in 3 na sliki Slika 2.4. Oponašanje pametne kartice napravi ali pametnem telefonu omogoča, da imamo podatke (običajno shranjene v pametni kartici) kar v telefonu. Ustrezna programska aplikacija s tem napravi omogoča plačevanje storitev, računov, služi pri javnih prevozih in podobno. Pri načinu P2P komunikacije se ustvari komunikacijski tok, v katerem obe napravi aktivno sodelujeta pri izmenjavi informacij. Ker ni potrebe po nastavljanju obeh naprav, se informacija prenese lahko zgolj z dotikom ekrana. Informacija lahko vsebuje datoteke, nastavitve, kontaktne podatke, sporočila in podobno.

2.4 Operacijski sistem Android

Android [12] je operacijski sistem, zasnovan na Linux jedru in razvit s strani podjetja Google. Slika 2.5 prikazuje uradni logotip. Operacijski sistem je pretežno namenjen mobilnim napravam in tabličnim računalnikom, vendar ga dandanes srečamo tudi v pametnih urah, televizijah, celo v avtomobilih, kamerah igralni konzoli Ouya [29] in drugih elektronskih napravah. Zaradi odprtokodnosti omogoča hitro in enostavno razvijanje aplikacij kot tudi prirejenih izpeljank samega operacijskega sistema. Najbolj popularen predelani Android sistem je CyanogenMod [13]. Sistem velja za enega najbolj razširjenih sistemov, poleg konkurenčnih Apple iOS in Windows Phone.

Slika 2.5: Logotip mobilnega operacijskega sistema Android.

Prva različica Android operacijskega sistema 1.0 je za javnost luč sveta ugledala 9. februarja 2009 [11]. Veljala je za zelo okrnjeno različico, ki je kljub temu vsebovala vse glavne funkcionalnosti telefona. Zanjo je kasneje izšla nadgradnja imenovana Android 1.5 oz. Android Cupcake. Nadgradnja je bila večja prelomnica za Android in je poleg standardnih funkcionalnosti dodala multimedijsko funkcionalnost, bluetooth povezljivost, animacije in podobno. Sledila je nadgradnja Android 1.6 in kmalu zatem Android 2.0. Vsaka od njih je prinesla tako novosti, kot tudi tehnične izboljšave pri delovanju naprav. Trenutno je na voljo že enajsta različica Android sistema imenovana Android Marshmallow – Android 6.0 ter Android 6.0.1. Javnost pa lahko preizkusi tudi že naslednjo različico Androida, tehnično poimenovano Android N. Ta različica je v preizkusni fazi in uradno še ni izdana. [12]

Za sistem Android obstaja ogromno aplikacij. Zaradi odprtokodnosti sistema, ki je glavni razlog za razcvet takšnih in drugačnih aplikacij, se nenehno izboljšuje uporabniška izkušnja in nasploh obogati sistem. Aplikacije so na voljo za prenos preko Googlovega spletišča Google Play.

2.5 Android aplikacija

Android aplikacija [22] je program, ki se izvaja na operacijskem sistemu Android. Pri razvoju aplikacije se uporablja programski jezik Java. Orodje za razvoj programske opreme kodo skupaj z ostalimi podatki pripravi v paket, ki ima končnico `.apk`. Tako pripravljen paket lahko nato namestimo na Android napravo. Aplikacija se po namestitvi izvaja v svojem posebnem okolju. V sistemu Android vsaka aplikacija predstavlja uporabnika. Vsak uporabnik od sistema dobi edinstveno številko. Prav tako ima vsak proces aplikacije svoj navidezni računalnik, ki poganja proces. S tem so procesi aplikacij izolirani med sabo. Za dodatno varnost ima aplikacija v danem trenutku dostop do sistemskih komponent le toliko, kolikor jih potrebuje. Aplikacija se začne izvajati, ko ena od njenih sestavnih komponent da znak za izvajanje, in se konča, ko je prenehala z izvajanjem, ali ko mora sistem sprostiti nekaj pomnilnika zaradi prezasedenosti.

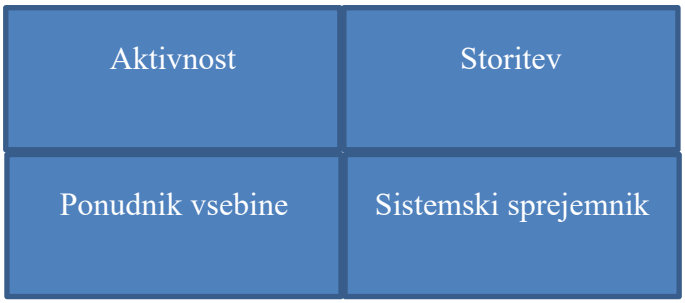
Aplikacijo sestavljajo štiri osnovni gradniki [14]. Slika 2.6 prikazuje spodaj opisane gradnike.

- **Aktivnost** (angl, activity) poskrbi, da se prikaže zaslon aplikacije skupaj z uporabniškim vmesnikom. Aplikacija se lahko sestoji iz več neodvisnih aktivnosti, ki programsko povezujejo zaslone aplikacij.

- **Storitve** (angl. services) omogoči izvajanje aplikacije v ozadju z namenom dolgoročnega izvajanja operacij aplikacije ali pa izvrševanje ukazov drugih procesov in aplikacij. Storitev z razliko od aktivnosti ne ponuja uporabniškega vmesnika.

- **Ponudnik vsebine** (angl. content provider) skrbi za podatke aplikacije. Te lahko prebere ali shrani na podlagi dostopnih pravic, ki jih ima aplikacija. Na voljo imamo zbirko v obliki ključ – vrednost, shranjevanje v datoteko ali shranjevanje v podatkovno zbirko. Ponudnik vsebine omogoča tudi spremembo podatkov ali dostop do podatkov preko drugih aplikacij, če to izrecno omogočimo.

- **Sistemski sprejemniki** (angl. broadcast receivers) delujejo nad celotnim sistemom. Gre za statusne signale, ki se prožijo, ko se denimo izklopi zaslon naprave, ko dobimo klic, ali sporočilo, ko je prazna baterija in podobno. Čeprav sistemski sprejemniki ne ponudijo uporabniškega vmesnika, imajo možnost ustvariti statusni opomnik, ki se prikaže v zgornjem delu ekrana v statusni vrstici in s tem uporabnika obvestijo o nekem dogodku. V splošnem je njegov glavni namen prehod iz enega gradnika na drugega. Pri tem naj bi opravil le manjša dela, denimo prožil inicializacijo storitve glede na sistemski dogodek.



Slika 2.6: Osnovni gradniki mobilne aplikacije.

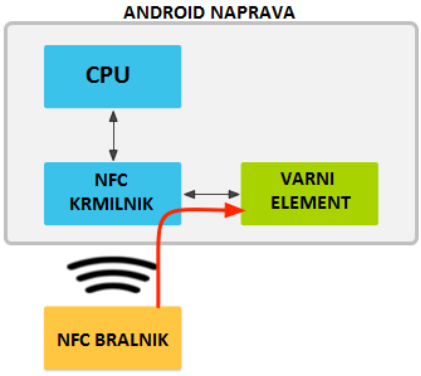
Android studio [24] je Googlovo orodje za razvoj Android aplikacij namenjenim pametnim telefonom, tablicam, pametnim uram in pametnim televizijam. Uporaba programa je brezplačna in je na voljo operacijskemu sistemu Windows, OS X in Linux.

2.6 Posnemanje kartic z uporabo pametnih telefonov

Veliko Android naprav, ki podpirajo NFC tehnologijo, podpira tudi posnemanje NFC kartice [21] (angl Host-Based Card Emulation ali HCE). Podpora posnemanja pametnih kartic s telefonom se je pri sistemu Android razvila v različici 4.4, kjer je poleg starejše metode

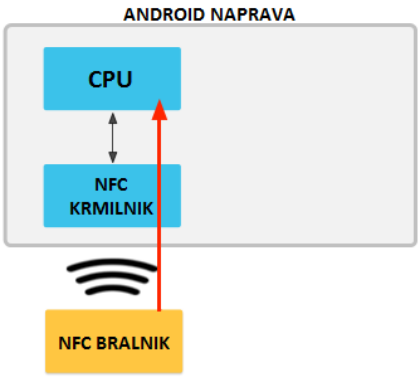
posnemanja kartice s pomočjo varnostnega elementa oz. SIM kartice telefona, pri tej verziji razvila dodatna metoda, ki za posnemanje kartice uporablja direktno NFC krmilnik v telefonu.

Pri metodi posnemanja kartice z uporabo varnega elementa se ob stiku telefona in NFC terminala ustvari komunikacijski tok, v katerem NFC krmilnik le posreduje podatke med varnim elementom in NFC terminalom. Pri tem se podatki ne obdelujejo s pomočjo krmilnika ali android aplikacije, pa tudi sama android aplikacija ni vključena v neposredno komunikacijo. Ko se komunikacija med varnim elementom in NFC terminalom zaključi, lahko android aplikacija pošilja poizvedbe varnemu elementu za nadaljnjo uporabo podatkov in obveščanju uporabnika. Slika 2.7 ponazarja opisano metodo.



Slika 2.7: Komunikacije NFC terminala in pametnega telefona z uporabo varnega elementa.

Pri metodi posnemanja NFC kartice brez uporabe varnega elementa se ob stiku pametnega telefona in NFC terminala vzpostavi komunikacija direktno z gostiteljem CPU, na katerem se izvajajo Android aplikacije. Slika 2.8 ponazarja opisano metodo.



Slika 2.8: Komunikacija NFC terminala in pametnega telefona brez uporabe varnega elementa.

HCE je zasnovana kot Android storitev. Dobra lastnost storitve je, da se brez interakcije uporabnika izvaja v ozadju samostojno. Pri plačevanju je to še posebej praktično, saj ni potrebe po zaganjanju aplikacij, ker se te ob stiku z NFC terminalom v ozadju zaženejo in samostojno izvedejo transakcijo med napravama. Za takšno preprosto delovanje v sistemu poskrbi proces izbire storitve – izbire aplikacije, s katero želi NFC terminal komunicirati ob stiku. Ko uporabnik tapne ali približa napravo NFC terminalu, mora sistem pametnega telefona izvedeti, s katero storitvijo želi komunicirati terminal.

Pri poizvedbi si pomaga s standardizirano metod ISO/IEC 17816-4 [19]. Ta definira način, kako izbrati pravo storitev na podlagi njene identifikacijske številke aplikacije – `AID`. Številko sestavlja 16 bajtov. Pri tem moramo paziti, da če naša storitev oz. aplikacija deluje po principu denimo plačevanja z znanimi karticami Visa, MasterCard, moramo v sistemu registrirati ustrezno že znano `AID` številko. V primeru, da želimo izdelati svojo storitev, ki bo uporabljala poljubno `AID` številko (ki ni javna oz. še ni znana), moramo paziti, da bo številka v skladu z specifikacijami standarda ISO/IEC 7816-5 [19]. Glavni razlog tiči pri preprečevanju dveh aplikacij za isto `AID` številko.

Da bo sistem prepoznal našo aplikacijo kot storitev, ki upravlja z NFC terminalom, jo moramo ustrezno deklarirati na posebnem mestu v `manifest` datoteki. To storimo z uporabo namenskega filtra in uporabo dovoljenja NFC modula.

```
<service android:name=".MyHostApduService" android:exported="true"
         android:permission="android.permission.BIND_NFC_SERVICE">
    <intent-filter>
        <action
android:name="android.nfc.cardemulation.action.HOST_APDU_SERVICE"/>
    </intent-filter>
    <meta-data
android:name="android.nfc.cardemulation.host_apdu_service"
          android:resource="@xml/apduservice"/>
</service>
```

Spodnja koda prikazuje deklaracijo HCE storitve aplikaciji ter registracijo `AID` številke naši storitvi. Slednja je zapisana v posebni XML datoteki, ki se mora začeti z značko `<host-apdu-service>`. Dodatno lahko določimo, ali je za delovanje HCE storitve potrebno napravo odkleniti ali ne. Vendar smo pri tem nekoliko omejeni. Ko izklopimo zaslon naprave, se posledično izklopi tudi NFC krmilnik, zato HCE storitev takrat ne deluje. Lahko pa deluje pri zaklenjenem zaslonu na podlagi naše deklaracije atributa `android:requireDeviceUnlock`.

```
<host-apdu-service
xmlns:android="http://schemas.android.com/apk/res/android"

    android:description="@string/servicedesc"
    android:requireDeviceUnlock="false">

    <aid-group android:description="@string/aiddescription"
               android:category="other">
        <aid-filter android:name="F0010203040506"/>
        <aid-filter android:name="F0394148148100"/>
    </aid-group>
</host-apdu-service>
```

2.7 Programsko okolje NetBeansIDE in paket WAMP

NetBeansIDE [15] je brezplačno in odprtokodno razvojno okolje izdelano v programskem jeziku Java. Čeprav je v glavnini namenjeno razvoju Javanskih aplikacij, NetBeansIDE omogoča razvoj aplikacij v drugih programskih jezikih kot so PHP, C/C++ in HTML5. Orodje ima zasnovano tudi storitev, ki omogoča uporabnikom izdelavo poljubnih modulov, ki še dodatno izpopolnjujejo ali pa implementirajo povsem novo funkcionalnost in tako še bolj pripomorejo k uporabniški izkušnji orodja. Module se lahko v orodje namesti neposredno preko vgrajenega čarovnika za nameščanje modulov ali pa si jih namestimo s pomočjo uradne spletne strani orodja NetBeans. Slika 2.9 prikazuje uradni logotip razvojnega okolja.

Slika 2.9: Logotip razvojnega okolja in platforme NetBeansIDE.

Windows, Apache, MySQL in PHP tvorijo celoto programskega paketa imenovanega WAMP [16]. Gre za odprtokodne programe, ki omogočajo delovanje spletnega strežnika. WAMP deluje le na operacijskem sistemu Windows. Programska oprema Apache služi kot spletni strežnik, MySQL kot podatkovni strežnik, kateremu je dodan še modul Phpmyadmin [25] za administracijo k podatkovni bazi ter PHP skriptni jezik, v katerem programsko povezujemo vse dele skupaj v celoto in ustvarjamo funkcionalnost strežnika. Slika 2.10 prikazuje uradni logotip strežniškega paketa.

Slika 2.10: Logotip Strežniškega programskega paketa WAMP

2.8 Javanski paket Java Smart Card I/O

Java SmartCardIO API [17] je javanski paket, ki omogoča komunikacijo pametne kartice in terminala po formatu sporočila APDU (*ang. Application Protocol Data Unit*) določenega po standardu ISO/IEC 7816-4. API vsebuje vse potrebne razrede (s funkcijami), s katerimi lahko izdelamo javanski program. Ta bo denimo poslal sporočilo APDU terminalu, priključenem na računalnik, terminal pa ga bo poslal kartici, ali drugi napravi, ki podpira NFC funkcionalnost in kot odgovor nazaj dobil sporočilo, v katerem bo neka informacija.

Obstajata dve vrsti tipa sporočila formata APDU [19], [20].

- **Ukazni APDU** (angl. Command APDU): ukaz pošlje naprava, ki želi v drugi napravi izvedeti določeno operacijo - branja podatkov, zapisovanja podatkov ali pa tekočo komunikacijo med napravama.

- **Odzivni APDU** (angl. Response APDU): ukaz pošlje druga naprava v komunikaciji kot odziv na ukaz prvotne naprave.

2.8.1 Ukazni APDU

UKAZNI APDU						
GLAVA - OBVEZEN DEL				TELO - POLJUBNI DEL		
CLA	INS	P1	P2	Lc	PODATKI	Le

Slika 2.11: Zgradba ukaznega APDU sporočila.

Slika 2.11 prikazuje zgradbo ukaznega APDU ukaza. Ta je sestavljen iz dveh delov. Obvezni del imenovan glava vsebuje bajte `CLA`, `INS`, `P1`, `P2`, Opcijski del imenovan telo pa ima bajte `Lc`, podatke ter `Le`. Polje `CLA` zaseda en bajt in pove vrsto ukaza, ki je specifičen za

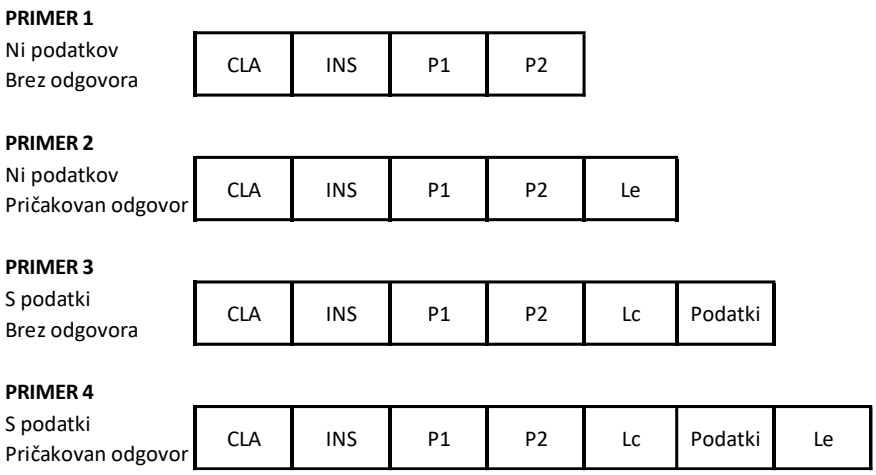
aplikacijo. Pri tem so vrednosti med `20` in `7F` rezervirane. Prav tako vrednost `FF`. Polje `INS` dolžine enega bajta točno določa za kateri ukaz znotraj vrste ukazov gre. Polje deluje v odvisnosti od polja `CLA`. Polja `P1` in `P2` dolžine enega bajta sta dodatna parametra polju `INS`. Lahko pa se uporabita tudi za zapis vhodnega podatka za potrebe aplikacije. Polje `Lc` označuje število bajtov, ki sestavljajo polje `Data Field`. Dolžina je določena s številom bajtov in sicer:

- če je dolžina `Lc` polja enaka 0, to pomeni, polje `Lc` ni vsebovan v ukazu, prav tako ukaz ne vsebuje polja podatkov,
- če je dolžina `Lc` polja enaka enemu bajtu, ki ima vrednost med 1 in 255, gre za kratko sporočilo, ki vsebuje polja `Lc` in podatki,
- če je dolžina polja enaka trem bajtom, mora prvi bajt biti nastavljen na vrednost 0, preostala dva pa določata dolžino podatkov med 1 in 65535 bitov.

Polje `Le` ima definirano dolžino do treh bajtov ali pa sploh nima definirane (0 bajtov). Polje definira največje možno število bajtov, ki se jih pričakuje v povratnem odzivnem APDU sporočilu.

Slika 2.12 prikazuje štiri tipične scenarije, ki se zgodijo med komunikacijo.

- V prvem primeru gre za ukaz, ki nima definiranih opcijskih podatkov, zato gre v tem primeru zgolj za izvedbo ukaza, ki bo nekaj spremenil, pobrisal, zamenjal, kopiral in podobno, zagotovo pa ne bo izvedel ukaza za branje.
- V drugem primeru gre za primer, ko ima ukazni APDU definirano le polje pričakovane dolžine bajtov odgovora, ki jih pričakuje kot odgovor na poslano sporočilo. Tipičen primer branja podatkov s pametne kartice.
- Pri tretjem primeru ima ukaz definirano dolžino vhodnih podatkov in dejanske podatke. Gre za primer, ko bo poslane podatke zapisal na kartico.
- Pri zadnjem primeru imamo polno definirano sporočilo. Ukaz bo z vhodnimi podatki izvršil definiran ukaz in v odgovor prejel željene podatke.



Slika 2.12: Vsi možni načini ukaznega APDU ukaza

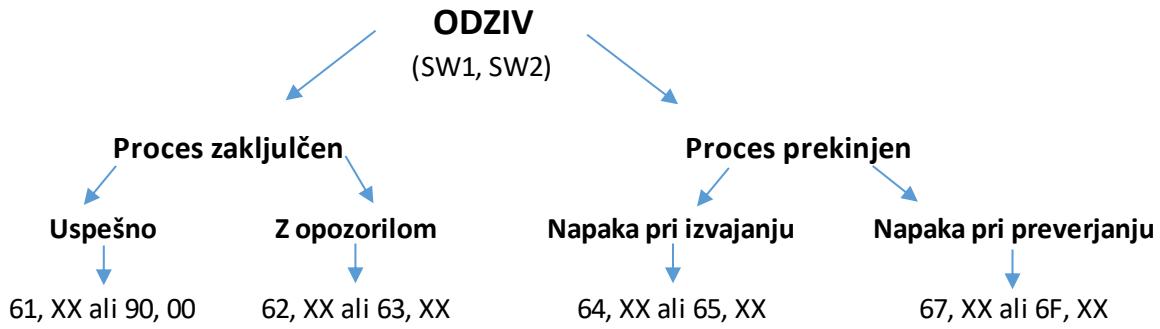
2.8.2 Odzivni APDU

Odzivni APDU je odgovor na ukazno APDU sporočilo in je prav tako sestavljeno iz dveh delov, vendar v drugačnem vrstnem redu. Slika 2.13 prikazuje sestavo odziva. Prvi del sestavlja telo sporočila in ni obvezen, drugi del pa sestavlja obvezen rep sporočila.

ODZIVNI APDU		
TELO - POLJUBNI DEL	REP - OBVEZEN DEL	
PODATKI	SW1	SW2

Slika 2.13: Zgradba odziva APDU.

Polje podatki so dolžine polja `Lc`, ki je določen in se nahaja v ukaznem APDU. Polja `SW1` in `SW2` sta dolžine enega bajta. Polji sta namenjeni za povratno informacijo o statusu prvotnega ukaza. Slika 2.14 prikazuje razvejanost statusa. Prvi del predstavlja uspešno razumljen ukaz. Pri tem gre lahko za popolno izvedbo ali pa izvedbo z opozorilom. Na drugi strani pa je prišlo do napake in posledično se ukazni APDU ne izvede. Do napake lahko pride pri samem izvajanju ukaza bodisi zaradi prekinitev komunikacijskega toka ali drugih dejavnikov, bodisi pa zaradi neskladja ukaza s standardom, nelogični sestavi ukaza ali pa nepravilnih parametrih.



Slika 2.14: APDU sporočila napak

2.9 JDBC API

Java Database Connectivity, krajše JDBC, API [18] omogoča delo s podatkovno bazo, tabelami, poizvedbami, sprožilci, in podobno za programski jezik Java. Vsebuje dva paketa in sicer paket `java.sql` in paket `javax.sql`. Prvi paket je namenjen dostopanju in obdelavi podatkov, ki so shranjeni v podatkovnih virih, običajno so to relacijske baze. Namen drugega je upravljanje in zagotavljanje dostopa do strežnika in podatkovnega vira.

Da lahko uporabljamo in izvršujemo funkcionalnost JDBC API-ja je potrebno zagotoviti tudi ustrezen gonilnik, ki mora temeljiti na enaki tehnologiji kot sama podatkovna zbirka. Namen gonilnika je prav to, da komunicira med podatkovno zbirko in ODBC API-jem.

Poglavje 3 Programska realizacija sistema

V poglavju podrobno opišemo nastanek prototipa. Podana je idejna zasnova, katero s pomočjo analize zahtev in definiranimi primeri uporabe dokončno realiziramo v delujoč prototip. Pri programskem delu so podrobneje predstavljeni in opisani posamezni programski deli.

3.1 Idejna zasnova

Cilj diplomske naloge je izdelati sistem, ki bo poskrbel za avtomatično izvajanje aktivnosti kontrole dostopa in beleženje prihodov in odhodov zaposlenih in pri tem omogočal še ročno popravljanje nepravilnega beleženja odhodov zaposlenih. Prototip sistema bo uporabo pametnih kartic nadomestili z uporabo pametnih telefonov.

Ker imamo dandanes v denarnici že cel kup kartic in so pametni telefoni čedalje bolj uporabni in ponujajo že vrsto vsemogočih funkcionalnosti, se je pojavila ideja o izgradnji prototipa sistema za prepoznavanje, ki bi jih nadomestil kar s pametnim telefonom. S tem bi zmanjšali količino kartic v naši denarnici, vendar se tudi v tem primeru ne bi mogli izogniti posledicam in nevšečnostim pri izgubi telefona.

Danes je podobnih programskih rešitev veliko in podjetja prodajajo tako sisteme prve kot druge vrste. Poleg tega ponujajo možnost še nadgradnje sistemov za specifične potrebe uporabnikov. Ker se zavedamo dejstva, da bi naš celovit sistem težko konkuriral današnjemu trgu takšnih sistemov, je glavna prednost v preprostosti uporabe in učinkoviti nastavitvi sistema, ki omogoča ter čim boljšo prilagoditev specifikam podjetja.

Prototip sistema sestavljajo:

- namizna aplikacija,
- mobilna aplikacija,
- podatkovna zbirka in

- NFC terminal.

Zahteve namizne aplikacije:

- ustrezna programska implementacija za izvajanje kontrole dostopa,
- beleženje prihodov in odhodov,
- dodajanje ali urejanje podatkov zaposlenih,
- komunikacija s podatkovno zbirko,
- komunikacija z NFC terminalom,
- izpis evidence ur in evidence kontrole dostopov in
- omogočanje ročnega popravljanja v primeru nepravilne uporabe funkcije odhodov.

Zahteve mobilne aplikacije:

- komunikacija z NFC terminalom,
- shramba prejetih informacij in
- pošiljanje podatkov na zahtevo.

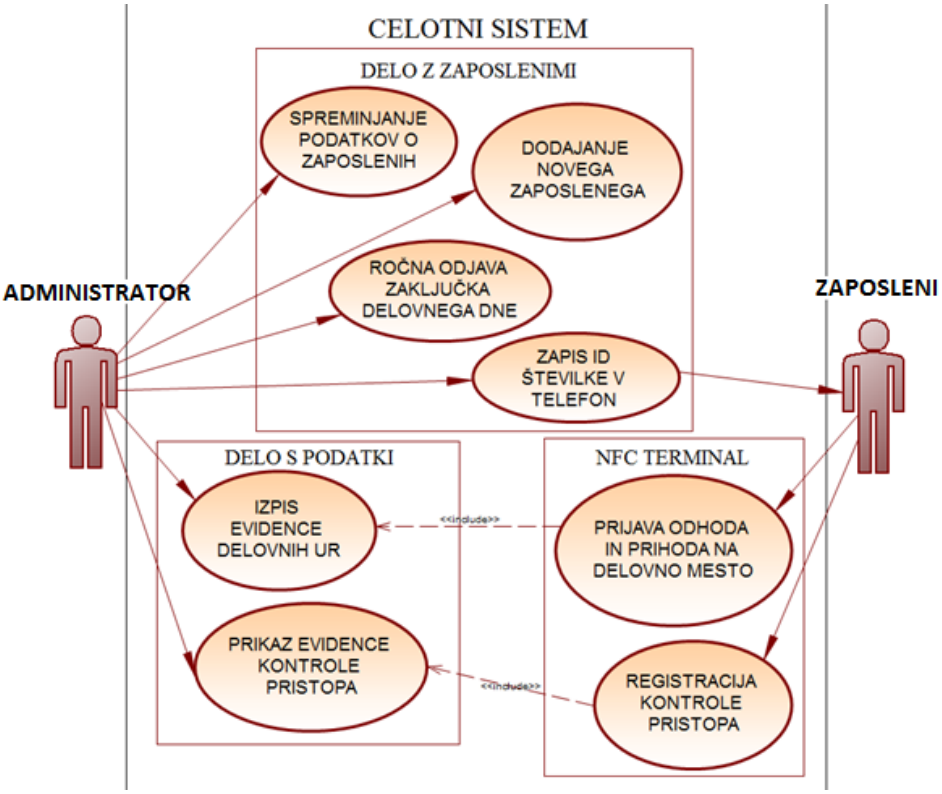
Zahteve podatkovne zbirke:

- shranjevanje podatkov in
- definiranje aplikacijskega uporabnika za namizno aplikacijo.

Zahteve NFC terminala:

- zagotavljanje komunikacijskega toka med namizno in mobilno aplikacijo.

Slika 3.1 prikazuje primere uporabe celotnega sistema in je namenjena dvema vrstama uporabnikov. Na levi strani je oseba (administrator), ki upravlja z aplikacijo, na desni strani pa je oseba (zaposleni), ki uporablja sistem. Administrator razpolaga s podatki zaposlenih in vodi evidenco ur, medtem ko se zaposleni v proces sistema vključi pri registraciji delovnega časa in kontroli dostopa.

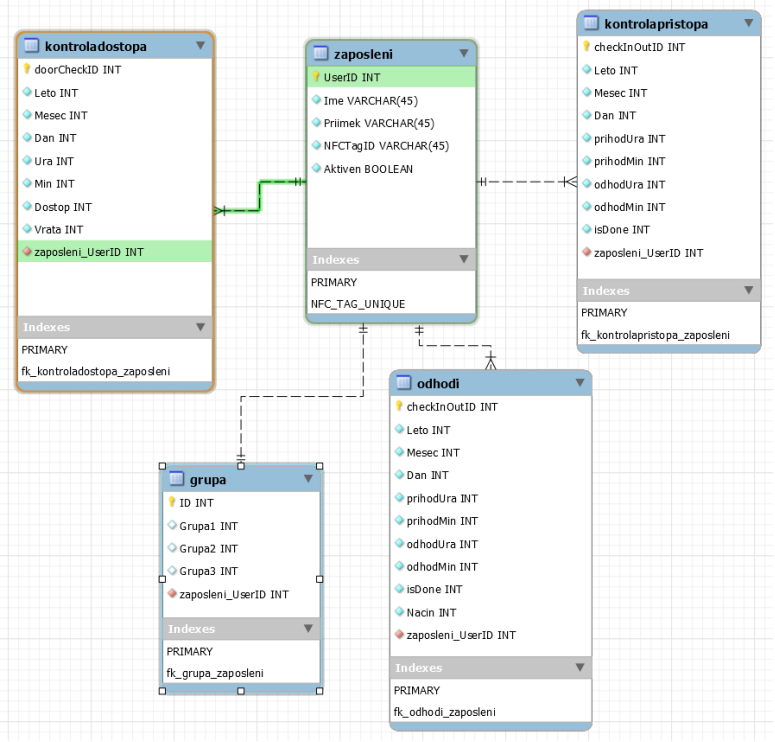


Slika 3.1: Primeri uporabe izdelani s pomočjo programa PowerDesigner.

Zasnove smo se lotili zelo sistematično in pregledno. Problem smo razbil na več manjših delov, za katere smo izdelali potrebno funkcionalnost. Ko je bila ideja po delih v celoti realizirana, smo se lotili programiranja vsakega dela posebej. Ko je bil določen kos v celoti sprogramiran, je sledilo še testiranje le-tega. Če je del uspešno prestal testiranje, smo ga dali na drug seznam, če je bilo to potrebno. Deli v drugem seznamu, so bili potrebni še grafične obdelave. V primeru, da del ni prestal testiranja, smo analizirali težavo in jo poskušali odpraviti.

3.2 Podatkovna zbirka

Podatkovna zbirka je sestavni del namizne aplikacije. Realizirana je s pomočjo strežniškega paketa WAMP, saj za vso tehnično delovanje poskrbi programski paket in smo se lahko takoj osredotočili na izgradnjo tabel, določevanje atributov ter njihovih pravil. Celotno podatkovno infrastrukturo smo realizirali s pomočjo modula Phpmyadmin [25], ki je del strežniškega paketa.



Slika 3.2: Relacijska shema podatkovne zbirke izdelana s pomočjo orodja MySQL Workbench.

Podatkovna zbirka vsebuje pet tabel: zaposlenih, pravice, registracija delovnega časa, registracija kontrole dostopa ter posebne odhode.

Dostop do podatkovne zbirke je omogočen z aplikacijskim uporabnikom. Gre za uporabnika, ki je nastavljen s pomočjo modula Phpmyadmin. Uporabnik ima pravice nad izvrševanjem operacij branja, pisanja ter spreminjanja podatkov v tabelah. Vse ostale pravice so onemogočene zaradi vidika varnosti.

Slika 3.2 prikazuje relacijsko shemo podatkovnega modela zbirke. Glavna tabela je tabela `zaposleni`. Vsak zaposleni ima mnogo zapisov prihodu in odhodu delovnega časa. Zapis sestavljajo atributi datuma, časa, enolične številke zapisa in tuj ključ, ki predstavlja zaposlenega. Prav tako ima zaposleni lahko več zapisov posebnih izhodov in kontrole dostopa, ki je po naboru atributov podobna tabeli odhodov z razliko o dodatnem atributu, ki določi način posebnega odhoda.

Zaposleni ima točno en zapis o skupini pravic. Zapis se hrani v tabeli `grupa`, ki jo sestavljajo trije atributi skupin. Vsaka skupina predstavlja dostopno pravico, na katero je pripet en ali več prostorov. Zaposleni ima lahko dostop do vseh treh skupin (upravlja najvišjo funkcijo) ali pa samo do ene (upravlja nižjo funkcijo).

Tabela `kontrola dostopa` lahko vsebuje mnogo zapisov o zaposlenem. Pri tem se ob vsaki sprožitvi preverjanja kontrole dostopa za posamezen prostor zabeleži podatek o zaposlenem ter čas in datum poizkusa dostopa in podatek o tem ali je dostop odobren oziroma zavrnjen.

3.3 NFC terminal

Za vzpostavitev komunikacijskega toka ter samo izmenjavo podatkov med namizno in mobilno aplikacijo skrbi ACR122U USB NFC bralnik/pisalnik podjetja Advanced Card Systems (ACS) [23]. Napravo prikazuje Slika 3.3.

Podjetje je specializirano na področju implementacije uporabe brezstične tehnologije, NFC tehnologije, mobilnih bralnikov pametnih kartic, izdelovanja programske opreme po meri in podobno.

Naprava ACR122U deluje na frekvenci 13.56 MHz in podpira bogato paleto brezkontaktnih kartic po različnih standardih. Te so Mifare kartice, standarda ISO14443 A in B, FeliCa pametne kartice in NFC značke. Naprava lahko s pametno kartico komunicira do hitrosti 424 kilo bajtov na sekundo in je skladna s PC/SC specifikacijo, ki je namenjena integraciji pametnih kartic v sistemsko okolje in poskrbi za komunikacijski protokol med napravo in pametno kartico. Ker ima sistemsko implementiran sistem za preprečevanje trkov, je na terminalu lahko prisotna v nekem trenutku le ena pametna kartica.

Naprava ima vgrajeno LED lučko, ki signalizira dogajanje med delovanjem. Prednastavljena rdeča barva pomeni, da terminal čaka na pametno kartico, ko je kartica prisotna, pa se lučka obarva zeleno. Poleg lučke pa ima v notranjosti pokrova še brenčač, ki ima enako nalogo kot lučka, le da odda zvočne signale. Oba elementa lahko programsko nastavimo glede na naše zahteve obnašanja.

Napravo se uporablja preko USB vrat in je za njeno delovanje potrebno namestiti ustrezen gonilnik. Slika 3.3 prikazuje izgled NFC bralnika, ki je uporabljen pri izdelavi prototipa aplikacije [23].

Slika 3.3: ACR122U NFC bralnik/pisalnik.

Poglavje 4 Programska implementacija sistema

V poglavju smo se lotili programske implementacije prototipa. Najprej smo izdelali namizno aplikacijo, nato še mobilno aplikacijo. Podrobneje so opisane nekatere funkcije namizne in mobilne aplikacije.

4.1 Namizna aplikacija

Namizno aplikacijo sestavlja več razredov. `NFCThread` in `MainWindow` razreda sta glavna, ostali pa so prisotni le pri določenih akcijah sistema. V glavni funkciji razreda `MainWindow` se nahajajo naslednje funkcije:

- `initComponents`(), katere naloga je inicializacija vseh elementov aplikacije,
- `resetNFCTerminal`(), katere naloga je vzpostavitev povezave med NFC terminalom in računalnikom preko USB povezave in morebitna ponovna vzpostavitev le-te, če med delovanju sistema pride do težav s terminalom,
- `setCustomBarColors`() poskrbi, da se privzete grafične vrednosti (tiste, ki jih preko grafičnega vmesnika ne moremo nastaviti) nastavijo poljubno in celotni aplikacij podajo izboljšano grafično podobo za boljšo uporabniško izkušnjo,
- `displaySimulatedTerminal`(), katere naloga je inicializacija pomožnega zaslona, ki je logično del NFC terminala.

Poleg funkcij vsebuje še nekaj ukazov, ki na začetku aplikacije vzpostavijo inicializacijski korak. To storimo z naslednjim izsekom programske kode.

```
BodyForm.setVisible(false);
ProstoriNastavitvePanel.setVisible(false);
jMenuNastavitve.setVisible(false);
jMenuItemPrikaziTerminal.setVisible(false);
jMenuItemPrikaziZemljevid.setVisible(false);
```

4.1.1 Glavna forma namizne aplikacije

Namizno zaslonsko formo `BodyForm` skupaj z meniji skrijemo in prikažemo z nastavitveno formo. Ta je sestavljena iz forme aplikacijskega uporabnika in forme nastavitve prostorov. Po končani interakciji prve forme se izvede programska koda, ki trenutno formo skrije in prikaže formo nastavitve prostorov. Poleg tega se izvede tudi funkcija, ki aplikacijo napolni s podatki iz podatkovne zbirke, npr. spustne sezname in podobno.

```
DriverManager.getConnection("jdbc:mysql://localhost:3306/Diploma",
username, sb.toString());
inicializirajPodatkeIzBaze();
ProstoriNastavitvePanel.setVisible(true);
SQLNastavitvePanel.setVisible(false);
```

Ko se zaključi še drugi del, programska koda poskrbi, da se trenutna forma ponovno skrije in prikaže glavni zaslon aplikacije. Ob zaključitvi inicializacije se dokončno inicializirajo in prikažejo gradniki, ki so odvisni od začetnih vpisanih parametrov. S pomočjo funkcije `osveziZemljevid`() se prikaže ločena forma, ki predstavlja shemo prostorov. Vsebina zemljevida je identična vsebini v koraku nastavitve prostorov. Forma čarovnika za nastavitev aplikacijskega uporabnika in prostorov se skrije in se prikaže glavna forma `BodyForm` skupaj z menijskimi izbirami. Prav tako se priče izvajati ločen proces, ki ga ustvari funkcija `NFCThread`. Proces skrbi za nenehno preverjanje značk na NFC terminalu.

Prav tako se v funkciji `startDailyTaskOdhodi`() ustvari ločen proces, katerega naloga je zbiranje obvestil in prikaz na posebno formo, ki je del `BodyForm`. Funkcija iz tabele `odhodi` na določen časovni razmik skrbi, da so vsi posebni izhodi pravilno zabeleženi, v smislu, da se je zaposleni pravilno prijavil dvojica posebnega odhoda in prihoda. Če se ugotovi, da je prišlo do nepravilnosti, funkcija izdela obvestilo in ga prikaže.

Glavna zaslonska forma je sestavljena iz dveh form. Na desni strani se nahaja forma z obvestili, na levi pa forma, ki vsebuje naslednje gumbe

- **Urejanje zaposlenih**, ki odpre novo okno za urejanje podatkov zaposlenih,
- **Odhodi**, ki deluje v povezavi z obvestili in omogoča, da administrator na podlagi prikazanih obvestil popravi dvojico posebnega izhoda in prihoda,
- **Evidenca ur**, ki prav tako odpre novo okno in omogoča statistični izpis evidence ur posameznega delavca,

- **Evidenca dostopa**, ki odpre novo okno in omogoča pregled dostopov za posameznega zaposlenega

4.1.2 Urejanje in spreminjanje podatkov zaposlenih

Forma urejanja zaposlenih je sestavljena iz vnosnih polj, spustnega seznama in potrditvenih polj. Vnosna polja služijo za vnos osnovnih podatkov potrditvena polja pa za določanje dostopa - pravic.

Pri vnosu novega zaposlenega se vrednosti izpolnjenih polj zabeležijo v ustrezno tabelo. Najprej se preveri ali gre za akcijo dodajanja ali pa zgolj spreminjanja podatkov. V primeru dodajanja se podatki vstavijo, v primeru spreminjanja pa posodobijo. Proces prikazuje spodnja koda

```
if(RBDodajZaposlenega.isSelected()) {
   // SQL Insert statement
   }
else {
   //SQL Update statement
}
```

Proces urejanja podatkov zaposlenih vsebuje še dodeljevanje edinstvene številke `NFCTagID`, ki se shrani v podatkovno zbirko in na mobilni telefon. Akcija je realizirana kot ločen proces, ki najprej zaradi varnosti in preprečitve sovpadanja procesa `NFCThread` le tega začasno ustavi. Nato se inicializira vzpostavitev med aplikacijo in mobilno napravo s pomočjo NFC terminala. Aplikacija ustvari kratko sporočilo po standardu ISO 7816-4 [19]. Sporočilo predstavlja `SELECT` ukaz. Funkcija `CommandAPDU`() kot vhodne parametre sprejme cela števila. Spodnja števila so v šestnajstiški obliki zapisana kot `0x00`, `0xA4`, `0x04`, `0x00`. Spremenljivka `AID_ANDROID` predstavlja enolično določeno identifikacijsko številko mobilne aplikacije, s katero želi namizna aplikacija komunicirati. Pri načrtovanju mobilne aplikacije je pomembno, da se ti dve številki ujemata.

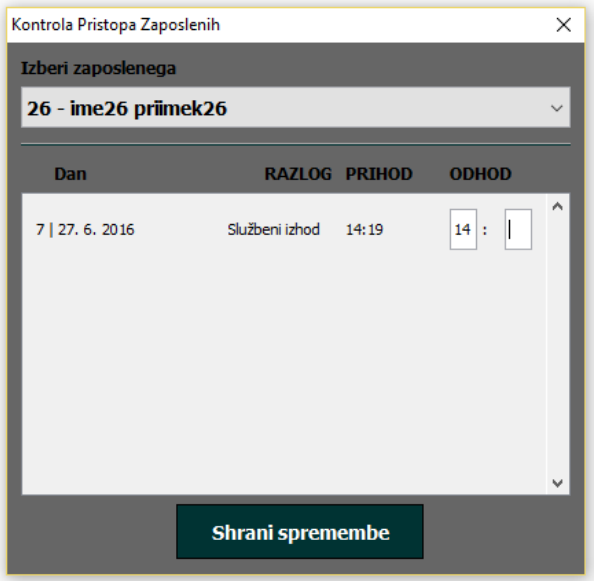
```
if(terminal.waitForCardPresent(5000)) {
   Card card = terminal.connect("*");
   CardChannel channel = card.getBasicChannel();
   apducommand = new CommandAPDU(0, 164, 4, 0,  AID_ANDROID);
   ResponseAPDU response = channel.transmit(apducommand);
   ...
}
if (response.getSW1() == 0x90 && response.getSW2() == 0x00) {
   ...
}
```

Po uspešno poslanem ukazu, se pričakuje takojšen dogovor mobilne aplikacije. Če je vzpostavitev uspešna, generiramo novo ukazno APDU sporočilo, v katerega kot podatek zapišemo dodeljeno edinstveno številko `NFCTagID`. Pri tem uporabimo ukaz za shranjevanje, ki je v šestnajstiški obliki definiran kot `0x00`, `0xDA`, `0x00`, `0x00`. Podatek `NFCTagID` se pred pošiljanjem za zagotavljanje višje varnosti, šifrira s pomočjo algoritma AES [31].

4.1.3 Ročno popravljanje evidence odhodov

Proces ročnega popravljanja posebnih izhodov zaposlenih v primeru nepravilne uporabe glavnega terminala je realiziran v posebni formi, ki jo sestavlja spustni seznam zaposlenih. Ob izbranem zapisu, se sproži funkcija, ki iz tabele odhodi za izbranega zaposlenega preveri, ali vsebuje kakšne nepravilne zapise. Zapis je nepravilen če je atribut `isDone` postavljen na 0. Za vsak nepravilen zapis se programsko ustvari nov razred, ki vsebuje oznako za čas prihoda, odhoda, oznako za način odhoda. Razredu se ustrezno nastavijo omenjene oznake glede na prebrano vrstico iz tabele. Na koncu se razred, ki je grafično realiziran kot panel doda v naprej pripravljen prostor forme. Slika 4.1 prikazuje grafično podobo podokna.

```
temp = New poprava Odhodov();
tmp.LPopravaDatum.setText(IDOdhod+" | "+dan+". "+mesec+". "+leto);
tmp.LPopravaPrihod.setText(odhodUra + ":" + odhodMin);
String mode = myRs.getString("Nacin");
if(mode.equals("1")) tmp.LMode.setText("Privatni izhod");
else if(mode.equals("2"))tmp.LMode.setText("Službeni izhod");
else tmp.LMode.setText("Malica");
```



Slika 4.1: Struktura podokna ročnega popravljanja posebnih odhodov.

Prazna vnosna polja na desni strani služita za vpis prihoda. Oba polja imata definirano preverjanje vnesenih znakov. Z validacijo vnosa se zagotovi, da pri popravljanju prihodov ne more priti do napačnih vnosov. Funkcija, ki prikazuje preverjanja vnosa za ure, je naslednja:

```
public boolean verify(JComponent input) {
  JTextField tmp = (JTextField) input;
  int stevilo;
  try {
    stevilo = Integer.parseInt(tmp.getText());
    return stevilo >= 0 && stevilo <= 24;
  } catch (Exception ex) {
    TFPrihodUra.setText("");
    return false;
  }
}
```

4.1.4 Izpis evidence ur

Tudi proces izpisa ur je realiziran na ločeni formi. Sestavlja ga spustni seznam in prostor, na katerem se prikaže statistika evidence ur za določenega zaposlenega. Ob izbiri zaposlenega se sproži funkcija, ki na podlagi podatkov iz tabele kontrole pristopa in odhodi pripravi podatkovno strukturo in jo ustrezno napolni. Najprej ustvarimo enostavno SQL poizvedbo, ki jo združimo po letih. Vsaka vrstica rezultata poizvedbe predstavlja posamezno leto. Nato za vsako leto ustvarimo zavihke. Zavihek je predstavljen kot nov razred `izpisLetnihUrPoMesecu()`. Grafično pa je predstavljen kot plošča, na kateri so oznake in sicer za vsak mesec ena. Vsaka oznaka ima definirano še funkcijo, ki se sproži, ko kliknemo na oznako. Tako inicializiran razred na koncu le še pripnemo kot zavihek.

```
while(myRs.next()) {
  izpisLetnihUrPoMesecih tmp = new izpisLetnihUrPoMesecih();
  jTabbedPane2.add(myRs.getString(2), tmp);
}
```

Podatkovna struktura je slovar, katerega ključi so cela števila, ki predstavljajo posamezen mesec, vrednost pa je še en ugnezden slovar. Notranji slovar ima kot ključ celo število, ki predstavlja zaporedni dan. Vrednost je tipa celo število in predstavlja trajanje delovnika tega dne.

```
Map<Integer, Map<Integer,Integer>> racunanje = new HashMap<>();
```

Ko je struktura napolnjena, je potrebno odšteti vse morebitne zamude prihodov malic, in privatne izhode. To storimo na podoben način. Ustvari se enaka struktura. Nato pa sledi odštevanje ur, če je to potrebno.

```
for(Integer mesec: racunanje.keySet()) {
  //<dan ,   trajanje> racunanje
  Map<Integer, Integer> poDnevih = racunanje.get(mesec);
  if(racunanjeOdhodi.containsKey(mesec)) {
    //<dan, trajanje>
    Map<Integer,          Integer>          poDnevihOdhodi          =
racunanjeOdhodi.get(mesec);
    for(Integer dan : poDnevih.keySet()) {
      int trajanjeDnevaRacunanje = poDnevih.get(dan);
      if(poDnevihOdhodi.containsKey(dan)) {
        int trajanjeDnevaOdhodiRacunanje = poDnevihOdhodi.get(dan);
        poDnevih.put(dan,          trajanjeDnevaRacunanje          -=
trajanjeDnevaOdhodiRacunanje);
      }
    }
  }
}
```

Koda ustrezno preračuna odbitke minut za posamezne dni. Na koncu poračunamo še skupno vsoto minut za posamezen mesec in vodimo evidenco za števil ur celotnega leta.

```
for(Integer mesec : racunanje.keySet()) {
  vsota = 0;
  Map<Integer, Integer> tmpDan = racunanje.get(mesec);
  for(Integer dan : tmpDan.keySet()) {
    vsota = vsota + tmpDan.get(dan);
    vsotaLeto += tmpDan.get(dan);
  }
  izracunPoMesecih.put(mesec, vsota);
}
```

Ostane nam še grafični prikaz. To storimo tako, da za vsako leto, ki je predstavljeno kot svoj zavihek, v ustrezne oznake, ki predstavljajo mesec, zapišemo število ur, ki smo ji izračunali iz zgornje strukture.

Na zaslonu terminala se nahajajo gumbi

- **Prihod** zabeleži čas prihoda na podlagi prebrane spremenljivke iz mobilnega telefona. Če nosi spremenljivka informacijo o posebnem izhodu, bo čas prihoda zabeležil v tabelo odhodov, drugače pa v tabelo kontrola pristopa.

- **Odhod** beleži klasične odhod ob zaključku delovnega dne.

- **Malica** v tabeli zabeleži čas posebnega odhoda in v telefon shrani spremenljivko, ki definira odhod na milico.

- **Službeni izhod** zabeleži čas odhoda v tabeli odhodi in v telefon shrani spremenljivko, ki definira službeni izhod.

- **Privatni izhod** zabeleži čas odhoda v tabelo odhodi in prav tako v telefon shrani spremenljivko, ki definira privatni izhod.

Zgoraj omenjena spremenljivka, ki se shrani v telefon, regulira ustrezne čase. Ob beleženju posebnega odhoda, spremenljivka dobi določeno vrednost, ki se ob prihodu ponastavi na privzeto vrednost. V primeru, da je v telefonu ostala shranjena vrednost, ki definira poseben izhod in zaposleni konča delovni dan, je prišlo do nepravilne rabe terminala. Ta je posledica pozabljene prijave prihoda od malice, službenega odhoda ali privatnega odhoda. V takšnem primeru se časi še vedno ustrezno zabeležijo. Zabeleži pa se tudi podatek, da je potrebno takšen primer rešiti ročno.

Ob pritisku na gumbe odhodov ali prihoda, se ustvari podproces, ki teče ločeno od glavnega procesa. V procesu se najprej počaka, da se mobilni telefon približa terminalu (največ nekaj sekund). Ko se telefon približa, se vzpostavi povezava s pomočjo `SELECT` ukaza. Nato pa se iz telefona prebere podatek o zadnjem odhodu, če gre za registracijo prihoda, ali shrani podatek o trenutnem odhodu, če gre za registracijo odhoda oz. posebnega odhoda. Pred nadaljnjo obdelavo hranjenega podatka ga je potrebno prej dešifrirati, pri shranjevanju podatka pa ga je potrebno šifrirati.

```
odhod(int mode) {
  Vzpostavi povezavo med aplikacijama s pomočjo NFC terminala
  Šifriraj podatek o odhodu
  Shrani podatek v telefon na podlagi spremenljive mode
  Shrani čas odhoda v ustrezno tabelo glede spremenljivke mode
  Prikaži obvestilo na zaslon terminala
}

prihod(Boolean opcija) {
  Vzpostavi povezavo med aplikacijama s pomočjo NFC terminala
  Preberi podatek o odhodu iz telefona
  Dekodiraj podatek
  Zabeleži prihod v ustrezno tabelo glede na podatek
  Prikaži obvestilo na zaslonu terminala
}
```

Psevdo kodi funkcij `odhod()` in `prihod()` sta predstavljani zgoraj. Pri obeh funkcijah se vzpostavi komunikacijski tok med namizno in Android aplikacijo s pomočjo NFC terminala. Pri funkciji `odhod()` se podatek o navadnem ali posebnem izhodu šifrira in zapiše v telefon. Prav tako se podatek o izhodu zapiše v ustrezno tabelo. Na koncu se s pomočjo zaslona terminala zaposlenega obvesti o uspešno izvedeni akciji. Funkcija `prihod()` deluje podobno

s to razliko, da podatek o odhodu, ki je zapisan na telefonu, najprej prebere in šele nato procesira. Na koncu se še v ustrezno tabelo zabeleži prihod zaposlenega.

4.1.5 Evidenca kontrole dostopa

Pri postopku kontrole dostopa, se zgodi naslednji dogodek. Najprej se iz označenega prostora na zemljevidu prebere, v kateri skupini pravic je definiran, nato se iz podatkovne zbirke pridobi podatek o dostopnih skupinah zaposlenega. Če pride do ujemanj skupin, se zaposlenemu vstop odobri, drugače pa zavrne.

Psevdo koda, ki je zapisana spodaj pridobi podatek o tem, v katero skupino pripada prostor, ki je izbran na zemljevidu. Nato program s pomočjo NFC terminala in mobilnega telefona zaposlenega pridobi podatek o dostopnih skupinah pravic zaposlenega. Če se ugotovi da zaposleni in prostor pripadata enaki skupini, je dostop odobren. Pri tem se obvestilo pojavi na zaslonu terminala. Če sta si skupini različni, se dostop zavrne in prav tako s pomočjo zaslona terminala obvesti zaposlenega. Na koncu se poizkus dostopa še zabeleži v ustrezno tabelo.

```
Za izbran prostor pridobi definirano skupino dostopa
Na podlagi zaznanega pametnega telefona na terminalu, pridobi
podatek o skupini zaposlenega
Primerjaj obe skupini
  če se ujameta
     Dostop odobren
  sicer
     Dostop zavrnjen
Zabeleži akcijo za evidenco kontrole dostopa
```

4.2 Mobilna aplikacija

Glavna naloga aplikacije je posnemanje pametne kartice. Android aplikacija je zato sestavljena zelo preprosto in sicer iz glavnega razreda `MainActivity` in pomožnega razreda `AppHCE`, ki razširja razred `HostApduService`. Omenjeni razred je obvezen, saj s tem mobilni aplikaciji omogočimo funkcionalnost komunikacije z NFC terminalom. Razred vsebuje metodi `processCommandApdu`() in `onDeactivated`(), ki ju je potrebno implementirati za delovanje.

```
processCommandApdu() {
  Preveri pravilnost obveznega dela ukaza
  Če je ukaz pravilno strukturiran
     Obdelaj podatke, zapisane v polju podatki ukaza APDU
     Obvesti o uspešnem procesiranju ukaza
     Vrni morebitne podatke
  Drugače pa vrni obvestilo o
```

```
      Nepoznani/nedovoljeni skupini ukazov (0x69, 0x81)
  Nepoznani podrobni inštrukciji ukaza (0x6D, 0x00)
}
```

Psevdo koda funkcije, ki skrbi za obdelavo sporočila iz namizne aplikacije, je predstavljena zgoraj. Ob sprejemu APDU sporočila se preveri, ali je ukaz pravilno zgrajen in se v primeru nepravilnosti izda odziv z napako. Če je ukaz zgrajen ustrezno, se izlušči podatkovno polje ukaza in se ga obdela. Pri obdelavi se preveri ali gre za ukaz branja ali pisanja. Po uspešnem ukazu se izda uspešen odziv skupaj z morebitnimi podatki, če so bili zahtevani.

Pri shranjevanju podatka v mobilni telefon se uporabi vmesnik `sharedprefference`. Pri shranjevanju se preveri tretji bajt shranjevalnega ukaza. V primeru, da je tretji bajt enak `0x00`, gre za shranjevanje enolične številke uporabnika. V primeru, da je bajt enak `0x01`, pa gre za shranjevanje posebnega odhoda.

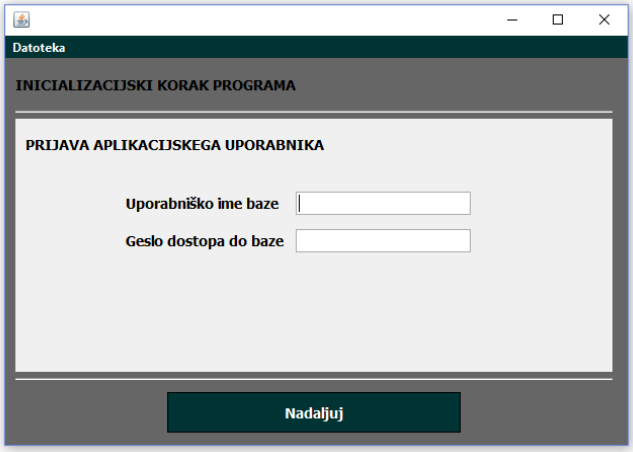
```
SharedPreferences settings =
PreferenceManager.getDefaultSharedPreferences(context);
SharedPreferences.Editor editor = settings.edit();

if(String.format("%02X", command[2]).equals("00"))
  editor.putString("ID", UID);
else if(String.format("%02X", command[2]).equals("01"))
  editor.putString("MODE", UID);
else
  return UNKNOWN_INS_SW;

editor.commit();
```

Poglavje 5 Analiza prototipa sistema

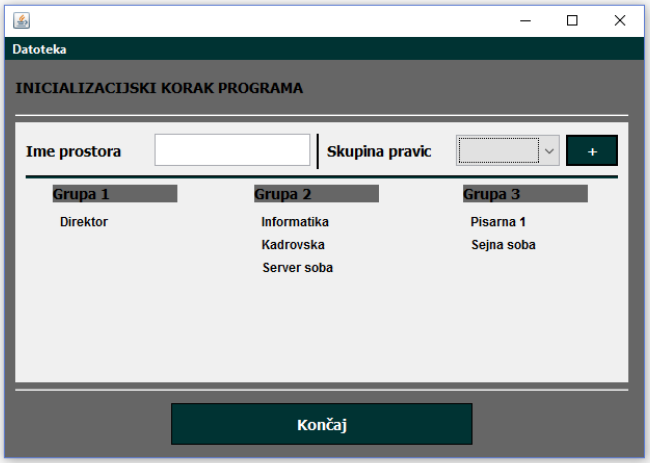
Končni prototip smo na koncu tudi preizkusili na testnih podatkih. Najprej smo s pomočjo namizne aplikacije ustvarili zgradbo prostorov in vpisali nekaj testnih zaposlenih oseb. Nato pa smo z mobilnim telefonom poizkušali dostopati do različnih prostorov in beležiti prihode in odhode.

5.1 Namizna aplikacija

Glavna aplikacija je namizna aplikacija. Sestavljena je iz inicializacijskega in osrednjega dela. Inicializacijski del, ki ga ponazarjata sliki Slika 5.1 ter Slika 5.2, predstavlja pripravo podatkov za pravilno delovanje glavnega dela, metem ko osrednji del že omogoča interakcijo z administratorjem. V osrednjem delu inicializacijskega postopka se najprej prikaže forma za vzpostavitev povezave s pomočjo aplikacijskega uporabnika in kasneje še forma, s katero definiramo fizične prostore podjetja in jih uvrstimo v posamezno kategorijo pravic.



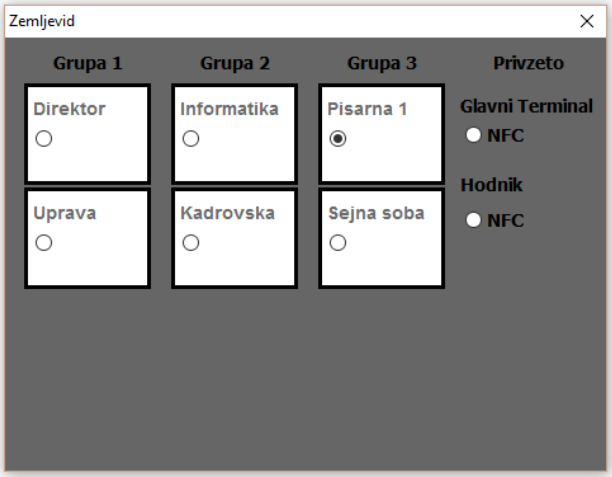
Slika 5.1: Inicializacijski korak – aplikacijski uporabnik.



Slika 5.2: Inicializacijski korak – uredba prostorov.

Sledi korak, ki je aktiven, dokler administrator ne zapre aplikacije oziroma dokler se ne zgodi napaka ali druga strojna okvara računalnika.

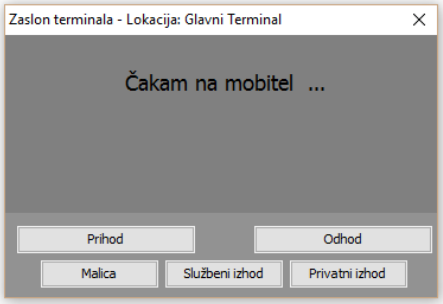
V zgornjem desnem kotu se pojavi samostojno okno, ki prikazuje razporeditev prostorov v posameznih skupinah pravic. Vsak prostor ima izbirni gumb. Ker smo pri prototipu imeli le en NFC bralnik, smo ga s klikom na željen izbrani gumb na zemljevidu, logično prestavili v izbran prostor. Slika 5.3 prikazuje grafično podobo izdelanega zemljevida prostorov.



Slika 5.3: Zemljevid prostorov podjetja

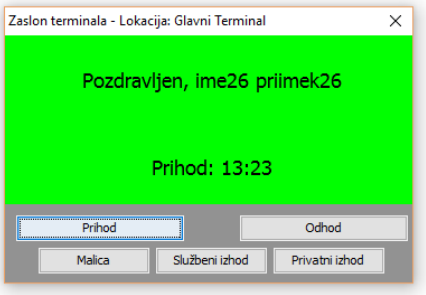
Če smo na zemljevidu podjetja izbrali prostor hodnik, je NFC terminal deloval kot kontrolnik vrat. Če smo ga logično prestavili v prostor namenjen glavnemu terminalu, pa je beležil prihode in odhode zaposlenih. Prav tako za potrebe demonstracije prototipa se v spodnjem

desnem kotu pojavi še zaslon terminala, ki ga program simulira v navezi z NFC terminalom. Slika 5.4 prikazuje zaslon terminala.



Slika 5.4: Zaslon terminala, logično postavljen v vhodnem prostoru podjetja.

Zaslon služi kot povratna informacija zaposlenemu, ko je ta v interakciji s terminalom, bodisi glavnim ali kontrolnim. V primeru, da je interakcija med zaposlenim, ki prihaja na delovno mesto, in glavnim terminalom, se na zaslonu izpiše ime in priimek zaposlenega ter ura prihoda. Slika 5.5 ponazarja prihod zaposlenega na delovno mesto.

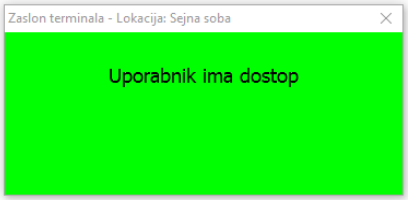


Slika 5.5: Zaslon terminala ob registraciji prihoda.

Ko zaposleni odhaja z delovnega mesta se na zaslonu terminala poleg imena, priimka izpiše še čas odhoda. Zaposleni dogodek prihoda ali odhoda določi s pritiskom na gumb, ki se nahaja na zaslonu, ko je NFC terminal logično postavljen poleg recepcije. V primeru logične postavitve NFC terminala v kateri koli drug prostor, se na zaslonu izpiše le ali ima zaposleni vstop v prostor odobren ali pa zavrnjen.

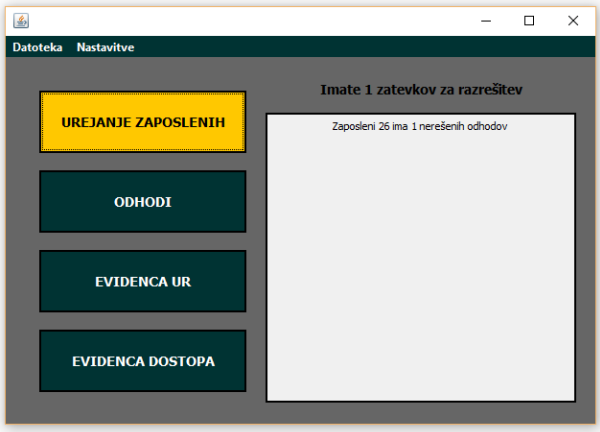
Dodatno se dostop prikaže v zeleni barvi, če je dostop potrjen ali rdeči barvi, če je dostop zavrnjen. Slika 5.6 prikazuje odobren dostop zaposlenega v nek prostor.

Zaslon vsebuje še tipke, ki definirajo posebne izhode: **malica**, **službeni** in **privatni izhod**.



Slika 5.6: Zaslon terminala logično postavljen v nek drug prostor podjetja pri uspešni kontroli dostopa.

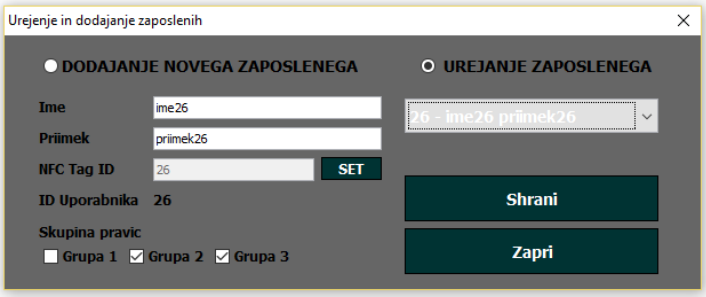
Slika 5.7 prikazuje glavni zaslon namizne aplikacije. Na levi strani glavnega zaslona aplikacije so gumb za različne dogodke, ki spreminjajo, ustvarjajo ali pa prikazujejo zapise iz podatkovne baze. Na desni strani je območje za obvestila. V omenjenem prostoru se prikazujejo sporočila, ki administratorja opozarjajo, da je pri beleženju časov prišlo do neskladja zaradi nepravilne uporabe terminala in podobno.



Slika 5.7: Glavno aplikacijsko okno.

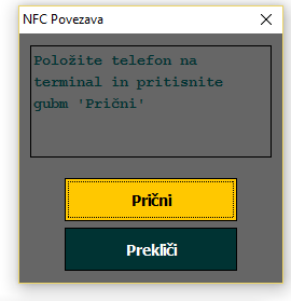
Administrator s klikom na gumb **urejanje zaposlenih** odpre podokno aplikacije. Slika 5.8 prikazuje grafični izgled podokna. Novo pojavna forma služi za ustvarjanje novega zaposlenega ali pa spreminjanje podatkov o zaposlenem uporabniku. V formi se zahteva, da so izpolnjeni vsi podatki, ki so:

- **ime**,
- **priimek**,
- **NFC Tag ID** polje in
- **skupina dostopa zaposlenega**, ki so programsko implementirane v tri sklope.



Slika 5.8: Primer uredbe podatkov o zaposlenih.

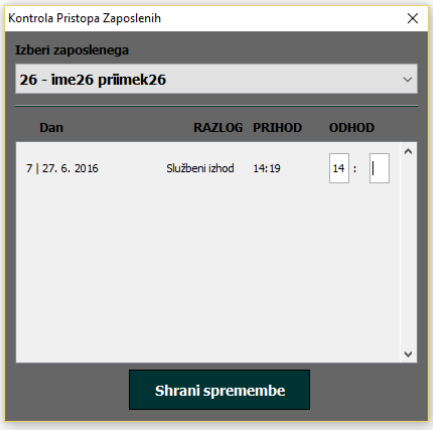
Pri določevanju enolične številke `NFCTagID` se ob kliku na gumb **set** prikaže novo pojavno okno. Slika 5.9 prikazuje grafično podobo podokna. Zaposleni mora NFC terminalu približati pametni telefon, administrator pa mora nato pritisniti na gumb **prični**. Program na podlagi identifikacijske številke zaposlenega ustvari enolično NFC številko in jo zapiše v podatkovno zbirko. Števila se zakodira in zapiše še v pametni telefon zaposlenega. Slika 5.10 prikazuje grafično podobo aplikacije po uspešnem zapisu podatka v pametni telefon. Administrator zaključi dodajanje ali spreminjanje zaposlenega, in ga s tem dokončno realizira še v podatkovni zbirki, s klikom na gumb **shrani**. V primeru izbire gumba **prekliči**, se administrator vrne v glavni zaslon aplikacije.



Slika 5.9: Zaslonska slika pred zapisovanjem podatka v pametni telefon.

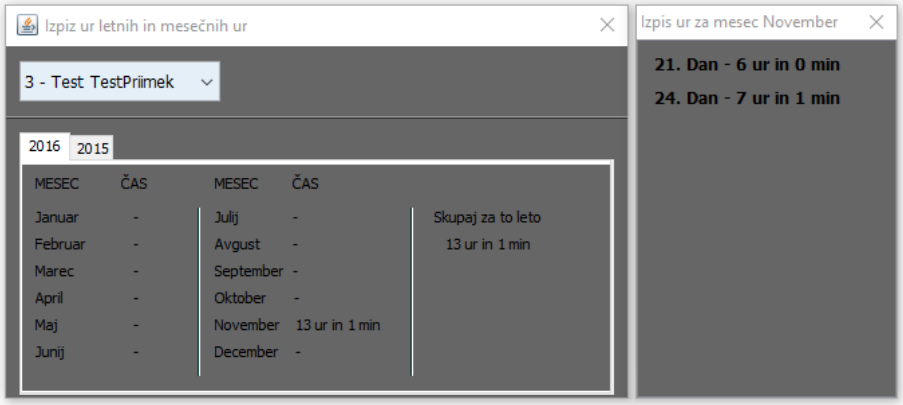
Slika 5.10: Zaslonska slika mobilne aplikacije po uspešnem zapisu podatka.

Gumb **odhodi** služi kot ročno popravljanje napak zaposlenih, v smislu, da zaposleni ob posebnih izhodih pozabi prijaviti prihod. S klikom na omenjeni gumb se odpre novo podokno aplikacije. Slika 5.11 ponazarja grafično podobo podokna, ki vsebuje spustni seznam, s katerega administrator izbere ustreznega zaposlenega. V spodnjem delu se nato prikažejo vsa neskladja posebnih odhodov. V posebej pripravljena polja mora administrator vpisati ustrezno uro in minuto prihoda. Da se zagotovi manjša možnost napake, obe polji preverjata pravilnost vnosa. S klikom na gumb **shrani** potrdi in popravi zapis v podatkovni zbirki.



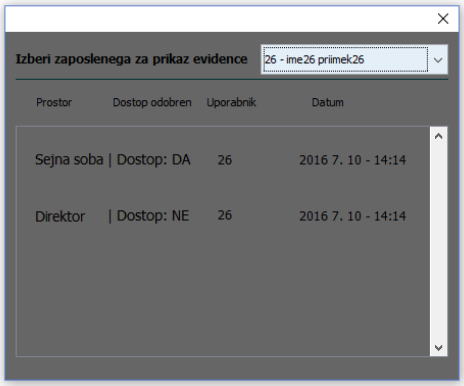
Slika 5.11: Ročno popravljanje prihodov posebnih izhodov.

S klikom na gumb **evidenca ur** ima administrator vpogled v evidenco delovnega časa zaposlenega. Odpre se novo podokno aplikacije, v katerem se nahaja spustni seznam. Slika 5.12 prikazuje grafično podobo podokna. S spustnega seznama se izbere zaposlenega in v spodnjem delu podokna se v zavihkih, ki predstavljajo posamezno leto, izpiše evidenca ur po mesecih. S klikom na posamezen mesec se v novem oknu prikaže še podrobnejši izpis ur po dnevih za izbrani mesec.



Slika 5.12: Zaslonski sliki izpisa evidence ur za leto (desno) in mesec (levo).

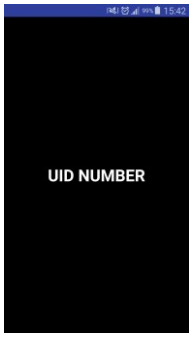
Zadnja izbira namizne aplikacije je statistični pregled kontrole dostopa. S klikom na gumb **evidenca dostopa** se odpre novo podokno. Ko administrator s spustnega seznama izbere zaposlenega, se v spodnjem delu prikaže seznam dostopov. Slika 5.13 prikazuje opisano podokno.



Slika 5.13: Zaslonska slika evidence kontrole dostopa,

5.2 Mobilna aplikacija

Mobilna aplikacija deluje posredno preko NFC terminala z namizno aplikacijo. Arhitektura aplikacije je zelo preprosta. Slika 5.14 prikazuje glavno zaslonsko sliko mobilne aplikacije. Njena glavna funkcionalnost je, da ob stiku z NFC terminalom posreduje enolično številko `NFCTagID` preko terminala v namizno aplikacijo, kjer se podatek naprej procesira. Za svoje delovanje moramo poskrbeti, da imamo vključeno NFC povezljivost, saj s pomočjo nje poteka komunikacija.



Slika 5.14: Zaslonska slika mobilne aplikacije.

Zaslonska slika glavnega in hkrati edinega zaslona aktivnosti, ki ga definira mobilna aplikacija. Tekstovno polje je namenjeno prikazu enolično določene številke, na podlagi katere se zaposleni identificira. Ta se ob procesu dodajanja novega zaposlenega prikaže v omenjenem tekstovnem polju.

5.3 Analiza opravljenega dela

V implementacijo celotnega prototipa, od ideje do končnega izdelka, je bilo vloženega veliko truda in znanja. Prav tako je svoj čas zahtevalo testiranje prototipa na testnih podatkih. Celotno izvajanje prototipa na testnem okolju je potekalo čez celoten teden, saj smo hoteli beležiti podatke za čim več različnih dni in različen čas. Pri testu je sodelovalo nekaj prijateljev. Ti so z različnimi verzijami Android sistema pridno uporabljali sistem.

V podatkovni bazi se je nakopičilo ogromno zapisov med izvajanjem našega poizkusa prototipa. Kar nekajkrat se je zgodilo, da se je pojavil napačen zapis v tabeli, ki pa je bil posledica napačne uporabe terminala oz. manjših tehničnih težav terminala. Zato je bilo potrebno terminal fizično ponovno vzpostaviti v sistem. Takšno napako smo nato odpravili tako, da se je NFC terminal programsko resetiral kar med delovanjem prototipa.

Prototip pravilno beleži vse prihode in odhode in ustrezno odšteva zamude. Komunikacija med aplikacijama deluje skoraj nemoteno in ne prihaja do izgub podatkov. Mobilni telefon ustrezno shranjuje podatke in posreduje informacije, ko to zahteva namizna aplikacija. Lahko bi rekli, da je sistem funkcionalno že nared za produkcijski test za neko konkretno podjetje.

Za umestitev prototipa v podjetje, ki ima nekje sto zaposlenih in nekaj več deset prostorov bi potrebovali večjo količino NFC terminalov. Vsak terminal bi se izvajal v svojem podprocesu, ločeno od glavnega procesa in bi za vsako interakcijo z zaposlenim beležil ustrezne čase in

kontrole dostopa v podatkovno zbirko. V primeru obsežnega številka zaposlenih, vključenih v sistem, bi razumljivo na dan izvedli ogromno zapisov.

Vsak prihod in odhod zaposlenega ter odhod in prihod malice bi predstavljal štiristo zapisov v tabeli. K tej številki dodamo še zapise o službenih prihodih in odhodih, ki smo jih ocenili, da bi se na dan pojavili povprečno trideset krat. To potem takem nanese še dodatnih šestdeset zapisov v tabelo. Pri tem pa ne pozabimo še na privatne izhode. Te smo ocenili, da bi se v povprečju lahko pojavili dvajset krat. Takšnih zapisov je potem štirideset.

Vseh zapisov bi bilo torej približno petsto na dan. Na teden bi jih naneslo okoli tri tisoč petsto in v celotnem mescu bi se nabralo okoli petnajst tisoč zapisov v podatkovno zbirko. Ker gre za povečano količino podatkov, je zato potrebna zadostna količina prostora, ki je namenjena shrambi podatkov in ustvarjanju varnostnih kopij.

Poglavje 6 Zaključek

V sklopu diplomskega dela je nastal delujoč prototip sistema kontrole dostopa in registracije delovnega časa. Ker so potrebe takšnega sistema zelo raznolike za posamezna podjetja, je skoraj nemogoče narediti nek univerzalni sistem, ki bi zadostoval vsem potrebam in pokril čim več podjetji. A vendar je prototip zasnovan tako, da pokrije oba sistema in ga je že mogoče uporabljati za produkcijsko okolje. Zaradi enostavnosti in preproste uporabe je prednost tudi to, da administrator prototipa ne potrebuje posebnega znanja za uporabo.

Delo je bilo zanimivo, predvsem pa zelo dinamično. Sistemi, ki temeljijo na NFC tehnologiji niso nova zadeva. Za identifikacijo uporabljajo pretežno pametne kartice. V diplomski nalogi pa samo naredili preskok in kartice zamenjali za pametne telefone, kar se je v praktičnem delu izkazalo za zelo dobrodošlo zamisel.

Prototip ima odprt širok spekter področij nadgradnje, nekatere izmed njih so naslednje:

- Spletna podpora, ki bi zaposlenim ponujala pregled ur ter upravljanja kadrovskih zadev, kot so dopusti, bolniške in podobno.
- Razširitev mobilne aplikacije, s katero bi zaposleni statistiko ur pregledoval praktično kjerkoli. Aplikacija bi ob stiku terminala prenesla podatke v telefon, zato nebi bilo potrebe po internetni ali podatkovni povezavi.
- Izboljšan grafični del za administracijo evidence prihodov, odhodov ter kontrole dostopa.
- Mesečno avtomatizirano razpošiljanje obvestil o delovnih urah zaposlenim.
- Nadgradnja terminalov še za uporabo naprav (tiskalniki, skenerji in podobno).
- Programska oprema za vpis zaposlenih na računalnik z uporabo pametnih telefonov.

Žal omenjene nadgradnje v prototipu zaradi časovne omejitve ter pomanjkanja NFC terminalov niso bile realizirane. Bodo pa zagotovo del naslednje verzije.

Literatura

[1] Implementing Biometric Time and Attendance Solutions to Increase Employee Productivity. [Online]. Dosegljivo: http://blog.m2sys.com/workforce-management/implementing-biometric-time-attendance-solutions-increase-employee-productivity/ [Dostopano 6. 7. 2016].

[2] Choosinga Time and Attendance System: 2016 Guide. [Online]. Dosegljivo: http://www.businessnewsdaily.com/6763-choosing-time-and-attendance-systems.html [Dostopano 6. 7. 2016].

[3] Kontrola pristopa. [Online]. Dosegljivo: http://www.mobicom.si/kontrola-pristopa [Dostopano 6. 7. 2016].

[4] Registracija delovnega časa. [Online]. Dosegljivo: http://www.kopa.si/glavnimenu/kopaerp/kopaerp/povezaneresitve/registracijadelovnegacasa.aspx [Dostopano 6. 7. 2016].

[5] Registracija delovnega časa. [Online]. Dosegljivo: http://jantar.si/sl/resitve.htm [Dostopano 7. 7. 2016].

[6] History of Biometrics [Online].]http://www.biometricupdate.com/201501/history-of-biometrics [Dostopano 7. 7. 2016].

[7] The Evolution of Access Control Systems. [Online]. Dosegljivo: http://securecomminc.com/2014/06/19/the-evolution-of-access-control-systems/ [Dostopano 7. 7. 2016].

[8] The History of Access Control. [Online]. Dosegljivo: https://www.ait.co.uk/blog/the-history-of-access-control [Dostopano 7. 7. 2016].

[9] NFC Skupnost. [Online]. Dosegljivo: http://www.nearfieldcommunication.org/ [Dostopano 7. 7. 2016].

[10] NFC, neprofitna organizacija. [Online]. Dosegljivo: http://nfc-forum.org/ [Dostopano 7. 7. 2016].

[11] Zgodobina OS Android. [Online]. Dosegljivo: https://www.android.com/history/ [Dostopano 7. 7. 2016].

[12] Android Operating System. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Android_(operating_system) [Dostopano 7. 7. 2016].

[13] Skupnost prilagojene različice Android OS. http://www.cyanogenmod.org/ [Dostopano 8. 7. 2016].

[14] Basic Application model. [Online]. Dosegljivo: https://developer.android.com/reference/android/app/package-summary.html [Dostopano 8. 7. 2016].

[15] Skupnost NetBeans. [Online]. Dosegljivo: https://netbeans.org/ [Dostopano 8. 7. 2016].

[16] WAMP. [Online]. Dosegljivo: http://www.wampserver.com/en/ [Dostopano 8. 7. 2016].

[17] SmartCardIO package summary. [Online]. Dosegljivo: https://docs.oracle.com/javase/7/docs/jre/api/security/smartcardio/spec/javax/smartcardio/package-summary.html [Dostopano 8. 7. 2016].

[18] JDBC. https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/ [Dostopano 8. 7. 2016].

[19] ISO7816-4: Interindustry Commands for Interchange. [Online]. Dosegljivo: http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4_5_basic_organizations.aspx [Dostopano 8. 7. 2016].

[20] An Intruduction to Java Card Technology – Part 1. [Online]. Dosegljivo: http://www.oracle.com/technetwork/java/javacard/javacard1-139251.html [Dostopano 9. 7. 2016].

[21] Host-Based Card Emulation. [Online]. Dosegljivo: https://developer.android.com/guide/topics/connectivity/nfc/hce.html [Dostopano 9. 7. 2016].

[22] Application Fundementals. [Online]. Dosegljivo: https://developer.android.com/guide/components/fundamentals.html [Dostopano 9. 7. 2016].

[23] ARC122U USB NFC Reader. [Online]. Dosegljivo: http://www.acs.com.hk/en/products/3/acr122u-usb-nfc-reader/ [Dostopano 9. 7. 2016].

[24] Android Studio. https://en.wikipedia.org/wiki/Android_Studio [Dostopano 9. 7. 2016].

[25] PhpMyAdmin. [Online]. Dosegljivo: https://www.phpmyadmin.net/ [Dostopano 24. 7. 2016].

[26] Google Wallet [Online]. Dosegljivo: https://www.google.com/wallet/ [Dostopano 24. 7. 2016].

[27] Apple Pay [Online]. Dosegljivo: http://www.apple.com/apple-pay/ [Dostopano 24. 7. 2016].

[28] Bluetooth [Online]. Dosegljivo: https://www.bluetooth.com/ [Dostopano 24. 7. 2016].

[29] Wi-Fi [Online]. Dosegljiov: http://www.wi-fi.org/ [Dostopano: 24. 7. 2016].

[30] OUYA [Online]. Dosegljivo: http://androidcommunity.com/ps4-xbox-one-ouye-an-ouya-copycat-20150818/ [Dostopano 24. 7. 2016].

[31] AES [Online]. Dosegljivo: http://aesencryption.net/ [Dostopano: 24. 7. 2016].

[32] Špica [Online]. Dosegljivo: http://www.spica.si/resitve/kontrola-pristopa [Dostopano: 1. 8. 2016].

[33] Četrta pot [Online]. Dosegljivo: http://www.cetrtapot.si/ [Dostopano: 1. 8. 2016].

[34] Jantar [Online]. Dosegljivo: http://jantar.si/ [Dostopano: 1. 8. 2016].